

---

## NAME

bp2fits - Converts bpipe format data to FITS format data.

## SYNOPSIS

**bp2fits** [*parameter=value*] [bp\_dpktf\_name=fits\_bintable\_col\_name]

## DESCRIPTION

**bp2fits** converts **bpipe** format data to FITS format data. As input, it takes a **bpipe** file(**input**) to convert and an RDB file(**trans**) that, among other things, provides a mapping between **bpipe** data packet field (dpktf) names and the FITS binary table column names. It outputs a FITS file containing a primary header data unit(PHDU), an additional header data unit(HDU), and a binary table. The PHDU contains cards coming from the RDB **trans** file. The HDU contains cards coming from the RDB **trans** file as well as the **bpipe** header data field(hdrf).

## OPTIONS AND ARGUMENTS

**bp2fits** uses an IRAF param style parameter interface.

The parameters are :

### input

The **bpipe** input stream/file. Set equal to `stdin` to read from the UNIX standard input stream.

### output

The FITS output stream/file. Set equal to `stdout` to write to the UNIX standard output stream.

### add\_exact\_dpktfs

Comma seperated list of **bpipe** data packet fields to add to the output FITS table. See *SELECTING FIELDS* for further info.

### add\_regex\_dpktfs

Comma seperated list of regular expression to match against **bpipe** data packet fields. Matches are added to the output FITS table. See *SELECTING FIELDS* for further info.

### del\_exact\_dpktfs

Comma seperated list of **bpipe** data packet fields to delete from the output FITS table. See *SELECTING FIELDS* for further info.

### del\_regex\_dpktfs

Comma seperated list of regular expressions to match against **bpipe** data packet fields. Matches are removed from the output FITS table. See *SELECTING FIELDS* for further info.

### trans

The file describing how to translate the **bpipe** data into FITS. This is used to massage the FITS into formats required by other programs.

If this is set to the string `default`, a wholly adequate default transformation is performed.

### longstrn

If set to yes, use the HEASARC OGIP Long String Keyword Convention. See [http://heasarc.gsfc.nasa.gov/docs/heasarc/ofwg/docs/ofwg\\_recomm/r13.html](http://heasarc.gsfc.nasa.gov/docs/heasarc/ofwg/docs/ofwg_recomm/r13.html)

### reserved

If set to yes, output various FITS reserved keywords to the HDU. See *Automatically Generated*

*Keyword Values* for more details.

**doublepass**

If set to yes and **reserved** is set to yes, write the **bpipe** once to determine values for TDMIN and TDMAX keywords from the data, then seek through and update the pertinent keywords. See *Automatically Generated Keyword Values* for more details.

**vectors**

If set to yes, represent **bpipe** vector data types as a single column in the FITS table. Otherwise, split each element of the vector into a separate column.

**help**

If true, it causes a brief synopsis of options to be written out.

**usage**

If true, it causes a extended explanation of **bp2fits** to be written out.

**verbose**

Level of debugging messages to be displayed.

**version**

If true, then the version number of the program will be written out and the program will exit.

## OVERVIEW

**bp2fits** converts a binary pipe format file (**bpipe**) to a FITS format file.

There are two distinct data structures in a a binary pipe format file, the header data field (hdrf) and the data packet field (dptkf). **bp2fits** attempts to create a one to one mapping between **bpipe** header fields and FITS HDU cards, as well as, between **bpipe** data packet fields and FITS binary table columns. By default, **bp2fits** will attempt to translate each hdrf and each dptkf in the input **bpipe** to a card and column, respectively, in the output FITS file. The user may modify the default translation or add keywords the the FITS header by specifying translation rules in the **trans** file, see *Trans File* for further info.

In general, the FITS standard is more restrictive with respect to the dimensionality of the data and the number of characters allowed to express values or field names. With this in mind, **bp2fits** will use the unmodified **bpipe** value or name where possible and will truncate the value or name where necessary. Setting the **verbose** flag will cause **bp2fits** to print warnings about many of the truncations and imperfect **bpipe** to FITS translations.

## SELECTING FIELDS

Four parameters control which **bpipe** data packet fields are passed on to the output FITS table.

**add\_exact\_dptkf**

**add\_regex\_dptkf**

**del\_exact\_dptkf**

**del\_regex\_dptkf**

If all four parameters are blank, all data packet fields are passed on to the output table. If one or both of the add\_\* fields are specified, a list of output columns is constructed based on the data packet fields which match the add\_list. Alternatively, if neither add\_\* field is specified, all input data packet fields are place in the add\_list.

If one or both of the del\_\* fields are specified, matches are removed from the add\_list.

Only the data packet fields remaining in the `add_` list after this process are written to output.

## CONVERTING BPIPE TO FITS

### Header Fields

**bpipe** header fields are named fields, possibly of multiple dimensions, of various data types. They correspond, roughly, to FITS HDU cards. As with FITS cards, there may be multiple header fields with the same name. Currently, each hdrf with the same name overwrites the previous hdrf when translating to FITS cards. Note, this does not apply to COMMENT or HISTORY keywords handed in through the **trans** RDB file.

There are a number of ways in which the two formats differ. Unlike FITS card keywords, **bpipe** hdrf names are not restricted to character strings of length less than or equal to 8 characters. Another issue in converting **bpipe** to FITS is that **bpipe** header fields may be multidimensional and FITS cards are always scalars. Finally, all possible **bpipe** hdrf data types do not map directly to FITS card value data types.

### Converting Header Field Names

As **bpipe** hdrf names are not restricted to the 8 characters that FITS card keywords are, we will outline the process where by **bp2fits** converts **bpipe** hdrf names to FITS HDU keywords.

- Any RDB header variables in the **trans** file are processed first.
- If a name is less than or equal to 8 characters long, it is used as is.
- If a name is greater than 8 characters long, the name is truncated so that the truncated name is 8 characters.
- If a truncated name collides with an existing name or another previously truncated name, it overwrites the previous name and value.

### Converting Header Field Data

The FITS standard allows for HDU cards to store the following data types: character string, logical, integer, floating point, complex integer, and complex floating point. This represents only a subset of the data types allowed in **bpipe** header fields. Below we list the allowed **bpipe** data types, and either the corresponding FITS card data type or a lack thereof.

**char** *Character String*

70 characters of ASCII text.

**double** *Floating Point Number*

Represented by as many as 20 characters.

**int** *Integer Number*

Represented by as many as 20 characters.

**uint** *Integer Number*

Represented by as many as 20 characters.

**DVector2** *No FITS Type*

First element is represented as a *Floating Point Number*.

**DVector3** *No FITS Type*

First element is represented as a *Floating Point Number*.

**IVector2** *No FITS Type*

First element is represented as a *Integer Number*.

**IVector3** *No FITS Type*

First element is represented as a *Integer Number*.

**UIVector2** *No FITS Type*

First element is represented as a *Integer Number*.

**UIVector3** *No FITS Type*

First element is represented as a *Integer Number*.

**DComplex** *Complex Floating Point Number*

First element is represented as a *Floating Point Number*.

**DCVector2** *No FITS Type*

First element is represented as a *Floating Point Number*.

**DCVector3** *No FITS Type*

First element is represented as a *Floating Point Number*.

In the cases where there is no analogous FITS type, **bp2fits** will warn the user if the **verbose** flag is set.

As **bpipes** header fields may be multidimensional and FITS cards are always scalars, **bp2fits** will copy the first element of a hdrf if possible and if the **verbose** flag is set, it will warn the user about the remaining elements.

## Data Packet Fields

**bpipes** data packet fields are named fields, possibly of multiple dimensions, of various data types. They correspond roughly to columns in FITS binary tables. As with FITS binary tables, a **bpipes** data packet contains fields with unique names. Unlike FITS binary column names, **bpipes** dptf names have no length restriction. Also, all possible **bpipes** dptf data types do not map directly to FITS binary table column data types.

## Converting Data Field Names

The FITS standard restricts the length of binary table column names to 70 characters. Since there is no such restriction placed on **bpipes** dptf names, **bp2fits** must handle names that are too long.

- Names which appear in the **trans** file are assigned the *fitsfield* value given there.
- Names which are less than or equal to 70 characters are used as is.
- Names which are greater than 70 characters are truncated.
- Names applied to vector data types

## Converting Data Field Values

When converting **bpipes** data packet field data to FITS binary table data, **bp2fits** follows this rule set:

**char** *Character* ASCII text.

**double** *Double Precision Floating Point*

64-bit double precision.

**int** *32-Bit Integer*

32-bit double precision.

**uint 32-Bit Integer**

Note that this is converted to a signed 32-bit integer. If the **verbose** flag is set, **bp2fits** will warn the user.

**DVector2 Double Precision Floating Point**

An array of double precision floating points will be used. The array indexing will be such that x,y pairs from the **bpipe** will be adjacent in the binary table and adjacent pairs in the **bpipe** will be off set by 2 in the binary table.

**DVector3 Double Precision Floating Point**

An array of double precision floating points will be used. The array indexing will be such that x,y,z triplets from the **bpipe** will be adjacent in the binary table and adjacent vectors in the **bpipe** will be off set by 3 in the binary table.

**IVector2 32-Bit Integer**

An array of integers will be used. The array indexing will be such that x,y pairs from the **bpipe** will be adjacent in the binary table and adjacent pairs in the **bpipe** will be off set by 2 in the binary table.

**IVector3 32-Bit Integer**

An array of integers will be used. The array indexing will be such that x,y,z triplets from the **bpipe** will be adjacent in the binary table and adjacent vectors in the **bpipe** will be off set by 3 in the binary table.

**UIVector2 32-Bit Integer**

An array of integers will be used. The array indexing will be such that x,y pairs from the **bpipe** will be adjacent in the binary table and adjacent pairs in the **bpipe** will be off set by 2 in the binary table.

**UIVector3 32-Bit Integer**

An array of integers will be used. The array indexing will be such that x,y,z triplets from the **bpipe** will be adjacent in the binary table and adjacent vectors in the **bpipe** will be off set by 3 in the binary table.

**DComplex Double Precision Complex**

An array of double precision floating points will be used. The array indexing will be such that r,i pairs from the **bpipe** will be adjacent in the binary table and adjacent pairs in the **bpipe** will be off set by 2 in the binary table.

**DCVector2 Double Precision Complex**

An array of double precision complex will be used. The array indexing will be such that c1,c2 pairs from the **bpipe** will be adjacent in the binary table and adjacent pairs in the **bpipe** will be off set by 2 in the binary table.

**DCVector3 Double Precision Complex**

An array of double precision complex will be used. The array indexing will be such that c1,c2,c3 triplets from the **bpipe** will be adjacent in the binary table and adjacent vectors in the **bpipe** will be off set by 3 in the binary table.

**Automatically Generated Keywords**

**bp2fits** will attempt to generate appropriate values for various column specific keywords. The user may override these using the **trans** file.

The following keywords are always generated: TUNIT, TNULL, TSCAL, TZERO, TDISP, TDIM.

The following are generated if **reserved** is set to yes.

#### TLMIN

This keyword is set to the value for TDMIN if not present in the **trans** file or if no **trans** file is specified and the **doublepass** flag is set. It is left out of the HDU if TDMIN is not present in the HDU.

#### TDMIN

This keyword is calculated from the data if not present in the **trans** file or if no **trans** file is specified and the **doublepass** flag is set. It is left out of the HDU otherwise.

#### TDMAX

This keyword is calculated from the data if not present in the **trans** file or if no **trans** file is specified and the **doublepass** flag is set. It is left out of the HDU otherwise.

#### TLMAX

This keyword is set to the value for TDMAX if not present in the **trans** file or if no **trans** file is specified and the **doublepass** flag is set. It is left out of the HDU if TDMIN is not present in the HDU.

### Trans File

The **trans** option specifies an RDB formatted file that guides the translation of the input **bpipe** file to the output FITS file. The **trans** file is required to have at least two columns, *bpipefield* and *fitsfield*. These columns provide a mapping between the input **bpipe** dpktf names and the output FITS binary table column names, see *Required Columns*.

In addition, the **trans** file may contain RDB header variables which are converted to FITS header keywords, see *Optional Header Variables*, and additional special columns described below, *Optional Columns*.

### Optional Header Variables

If the user wishes to add FITS header keywords to either the primary HDU or the HDU of the binary table in the output FITS file, it is possible to do so via the optional RDB header variables in the **trans** file.

As described in *CONVERTING BPIPE TO FITS*, **bp2fits** handles converting **bpipe** header fields to FITS header keywords. By specifying RDB header variables in the **trans** file, the user may add FITS keywords to the output. By default, **bp2fits** places RDB header variables in the HDU associated with the FITS binary table. It is also possible to force the header variables in the FITS file's primary HDU. This is done by prepending the modifier *PRIMARY::* to the RDB header variable name.

```
...snip...
#: PRIMARY::IWRITETO=Primary Header Data Unit
#: IWRITETO=The binary tables HDU...
...snip...
```

Certain FITS header keywords call for special values, such as the date the file was created, or the name of the user who created the file. To enable the user to handle these cases, **bp2fits** provides the user with special header field values denoted by a leading % character.

### Special Header Field Values

**bp2fits** defines three special header field values which may be specified in the **trans** file.

**%user**

If a header variable in the **trans** file is set to **%user**, **bp2fits** attempts to determine the current user name via first `getlogin` or, failing that, `getpwuid`.

**%date**

If a header variable in the **trans** file is set to **%date**, **bp2fits** attempts to determine the current date via the `localtime` function. The date is output in day/month/year format.

**%time**

If a header variable in the **trans** file is set to **%time**, **bp2fits** attempts to determine the current time via the `localtime` function. The time is output in hour:minute:seconds format.

**Required Columns**

The **trans** file must contain at least two columns, *bpipefield* and *fitsfield*. The *bpipefield* and *fitsfield* columns allow the user to explicitly map the **bpipe** dptf name to a valid FITS binary table name. If a **bpipe** dptf name is not listed in the **trans** file *bpipefield* column, it is treated in the default manner by **bp2fits**, see *CONVERTING BPIPE TO FITS Data Fields*.

**bpipefield**

Dptf name in **bpipe**, exact match required.

**fitsfield**

FITS binary table column name. Must be an allowable FITS value, i.e. less than 70 characters, comprised of letters, digits, and the underscore character `_`.

**bpipe** vector fields are mapped to multiple FITS binary table columns. The user may specify the column names as follows:

- A common prefix, to which the default suffixes (which depend upon the actual **bpipe** data type) are appended, e.g. `position_` which in the case of a **DVector3** would yield `position_x`, `position_y`, and `position_z`.
- A common prefix and user supplied suffixes, e.g. `position_x,YYY,Zeh` which would yield `position_x`, `positionYYY`, and `positionZeh`.
- The special character `!` and user supplied suffixes, e.g. `!_x,YYY,Zeh` which would yield `_x`, `YYY`, and `Zeh`.

**Optional Columns**

There are certain optional FITS keywords that are useful to add to the output header, e.g. `TUNIT`, `TNULL`, `TSCALE`, etc. In many cases, it is difficult to determine appropriate values from the input **bpipe**. **bp2fits** allows the user to supply values for various FITS reserved keywords via the **trans** translation file.

The **trans** file may contain any of the following, optional, columns as well: `units`, `null`, `scale`, `zero`, `disp`, `dim`, `lmin`, `dmin`, `dmax`, or `lmax`. These columns correspond to the indexed FITS keywords of the same name, e.g. `lmin == TLMINxxx` where `xxx` is the index of the column `TLMIN` refers. If a **bpipe** data packet fields's optional translation columns are not specified, it is treated in the default manner by **bp2fits**, see *CONVERTING BPIPE TO FITS Data Fields*.

A listing of the optional columns is provided below:

**units**

The physical units for the corresponding field.

**null**

The integer that represents an undefined value for fields of type B(unsigned 8-bit integer), I(16-bit integer), or J(32-bit integer).

**scale**

The scale factor for the corresponding field to scale it to a physical quantity, whose possible units are specified via the **units** column. Used along with **zero**. **scale** is ignore for fields of type A(Character), L(Logical), or X(Bit Array).

**zero**

Used along with **scale**, it is the zero point for the physical representation of the field value. **zero** is ignore for fields of type A(Character), L(Logical), or X(Bit Array).

**disp**

Character string describing the format recommended for displaying the data in the corresponding field.

**lmin**

The legal lower limit for data in this field.

**dmin optional**

The actual minimum value for the data in this field.

**dmax optional**

The actual maximum value for the data in this field.

**lmax**

The legal upper limit for data in this field.

## VERSION

This documents version 1.2.4 of **bp2fits**.

## COPYRIGHT AND LICENSE

Copyright 2006 Smithsonian Astrophysical Observatory

This software is released under the GNU General Public License. You may find a copy at

<http://www.fsf.org/copyleft/gpl.html>

## AUTHORS

Diab Jerius <djerius@cfa.harvard.edu>

Michael Tibbetts <mtibbetts@cfa.harvard.edu>