A Hierarchical Grouping Convention for FITS

Donald G. Jennings, ISDC
William D. Pence, GSFC
Michael Folk, NCSA
Barry M. Schlesinger, GSFC/HSTX

Proposal Draft, Revision 8
June 12, 1997
Minor updates: May 2007

Abstract

This paper describes a grouping convention for FITS that facilitates the construction of hierarchical associations of Header Data Units (HDUs). The grouping convention uses FITS table structures (ASCII or binary) to encapsulate pertinent information about the HDUs belonging to a group. Group members may reside in a single FITS file or be distributed in many FITS files; the FITS files themselves may reside on different computer systems.

1 Introduction

The rules for generalized extensions in FITS (Grosbøl *et al.*, 1988) provide for FITS formatted files containing more than one header data unit. By using combinations of ASCII tables (Harten *et al.*, 1988), binary tables (Cotton *et al.*, 1994) and image extensions (Ponz *et al.*, 1994) related data sets requiring different data structures may be stored in the same FITS file, each within its own HDU. Unfortunately, once the related data sets are segregated into separate HDUs the relationship between them is often lost.

The FITS standard currently allows for simple hierarchical associations of HDUs within a single FITS file through use of the EXTLEVEL keyword. However, this mechanism has several major limitations. First, its use is not well defined. Different organizations may use EXTLEVEL for widely varying purposes and still not violate the FITS standard. Secondly, it does not specify a mechanism for defining distinct multiple *groups* of HDUs within a FITS file. Lastly, it cannot be used to associate HDUs residing in different FITS files. Except for very simple cases, FITS contains no mechanism for creating or preserving associations between HDUs or groups of HDUs.

As the volume and complexity of FITS formatted data grows, the need for a recognized and versatile HDU grouping mechanism increases. Individuals can be overwhelmed trying to manage and analyze large data sets unless those sets are logically organized. Software tools also require data organization in order to access all necessary components of an observation, simulation or experimental data set.

As an example of where grouping capabilities within FITS would be useful, consider the following. It is desirable to combine a set of observations from a given time period into a single

FITS file for transport and archival purposes. For each observation there is an observation log, an event list, a derived image and a set of instrument calibration data; furthermore, several observations share a common set of calibration data. By using a grouping mechanism each [log, event list, image, calibration] set could be logically tagged as an associated observation group and the calibration data could be made a part of many different observation groups, thus eliminating the need to store it more than once. Software could retrieve all the information about a given observation simply by extracting those HDUs defined in the table that identifies members of the group. Also, observations of the same object from different observational periods could be combined into a group and accessed as a unit, even though the HDU sets comprising the different observations reside in separate FITS files.

The following sections describe a scheme for implementing a hierarchical grouping of header data units within single and multiple FITS files. Section 2 discusses the content of table extensions used to define HDU groupings. Section 3 lists those keywords recommended for headers of group member extensions. Finally, Section 4 provides sample headers from FITS table extensions containing grouping structures.

2 Group Tables

A *group table*, as defined in this convention, is a FITS table extension that contains a list of all the associated member HDUs in the group. Group tables may be represented by either FITS ASCII tables (`XTENSION=␣'TABLE␣␣␣'`) or binary tables (`XTENSION=␣'BINTABLE'`), and are uniquely distinguished from other types of FITS tables by having the `EXTNAME␣=␣'GROUPING'` keyword and value in the header. The other required or recommended keywords and columns in a group table are described in the following sections.

There may be zero, one, or more group tables within a given FITS file. Each group table may reference any number of HDUs. The entire set of HDUs referenced in a group table, along with the group table itself, form a *group* . Individual HDUs referenced in a group table are said to be *members of the group* or *group members*.

Groups can contain any type and mix of HDU. This includes all of the IAU-endorsed extensions as well as other extensions that conform to the requirements for generalized FITS extensions. Note that a group may also contain other groups as members, since a group table is itself a FITS extension. This feature allows for the construction of hierarchical structures of HDUs within a single FITS file or across many FITS files. Recursive grouping, where a grouping table either directly or indirectly includes itself as a group member, is expressly forbidden. While it is possible for a HDU be included more than once within a group, this usage is not recommended.

2.1 Group Member Identification Methods

Group tables specify the names and locations of FITS files containing member HDUs as well as identifying members within their FITS files. The name and location of each FITS file is specified by using the World-Wide Web (Berners-Lee, 1994) Uniform Resource Identifiers, or URIs. All current and future forms of URIs, such as Uniform Resource Locators (URL) and the proposed Uniform Resource Names (URN), shall constitute valid names, although the group table must specify the type of URI being used. If the group member resides in a different FITS file but on the same computer system then partial URIs (specifically partial URLs) may be used instead of absolute URIs to specify the member's file location. If the group member resides in the same FITS file as the group table itself, then the URI field may be left blank.

Since the location of external HDUs may change (e.g., the file may be deleted or moved to a different location), software that attempts to access external member HDUs of a group should be prepared to deal with the possibility that the file no longer exists.

The location of member HDUs within FITS files may be specified in two different ways, either by *reference* or by absolute *position*. The reference identification method uses the values of the XTENSION, EXTNAME and EXTVER keywords to uniquely identify the member HDU within the FITS file. The position method uses the HDU order number to identify members, with the primary array having order value 0, the first extension order value 1, and so on. Users may choose either or both identification methods when constructing a group table.

While the reference method is not invalidated by a reordering of HDU positions within FITS files, it does require that each member HDU have a unique set of (non-FITS-required) keyword values. Thus, this method may present problems for FITS files whose headers cannot be easily modified, such as FITS files on read-only media. The position identification method provides for quick "random" access to the member HDUs, since software does not have to sort though each extension looking for the correct set of keyword values, but will be affected if the order of member HDUs within their FITS files is changed (please note: there is nothing within the current FITS standard governing how or when HDUs may be reordered within their files).

2.2 Group Table Keywords

In addition to the standard required FITS table extension keywords, the following keywords are required in the header of a group table:

- **EXTNAME (character)**: This value of the FITS reserved keyword uniquely identifies that this FITS extension contains a group table. For group tables EXTNAME must have the value 'GROUPING'.
- **EXTVER (positive integer)**: The value of this FITS reserved keyword serves as a group ID number that uniquely distinguishes this group from any other groups that may be defined in the same FITS file. All HDUs in a given FITS file with `EXTNAME␣=␣'GROUPING'` must have a unique integer EXTVER value. This group number may also be used in the header of each group member to identify the group(s) to which the member belongs (see section 2.3, GRPIDn keyword). While it may be convenient to assign consecutive EXTVER values, starting with 1, to each grouping table that is defined within a file, this is not required.

The following keyword is strongly recommended for inclusion in the header of each group table:

- **GRPNAME (character)**: This keyword contains the name associated with the group table. GRPNAME values are case-insensitive and should only contain letters, digits, and the underscore character (and not contain any embedded blank (ASCII 32) characters).

2.3 Group Table Columns

The number of columns required in a group table depends on which method is used to identify the members (and recall that both methods may be used within the same group). If the members are identified by reference then the following columns are required:

- `TTYPEn␣␣=␣'MEMBER_XTENSION'` – **character field:** Contains the value of the XTENSION keyword from the group member's header. In the case of primary HDUs where there is no required XTENSION keyword, the value of 'PRIMARY' will be used instead. Therefore, the current valid entries for this column are `'PRIMARY␣'`, `'TABLE␣␣␣'`, `'BINTABLE'`, `'IMAGE␣␣␣'` or any other IAU FITS Working Group registered XTENSION value. Note that the single quotation marks are used only to designate the string boundaries and are NOT to be included with the XTENSION values in the column entries; the trailing blanks shown in each string are optional. This field may contain the FITS null value appropriate for this column type if the value is unknown (e.g., if the position identification method described below is used to identify the member location).

- `TTYPEn␣␣=␣'MEMBER_NAME'` – **character field:** Contains the value of the EXTNAME keyword from the group member's header. In the case of primary HDUs where the EXTNAME keyword is not defined or when the member extension has no EXTNAME keyword present, this field may contain the FITS null value appropriate for the column type.

- `TTYPEn␣␣=␣'MEMBER_VERSION'` – **integer field:** Contains the value of the EXTVER keyword from the group member's header. In the case of primary HDUs, or if the EXTVER keyword is not present in the member header then a value of 1 should be assumed.

If members are identified by file position then the following column is required:

- `TTYPEn␣␣=␣'MEMBER_POSITION'` – **integer field:** Contains a group member's position within its FITS file. The file's primary header is given a position value of 0, the first extension is given a position value of 1, and so on. If for some reason a group member's 'MEMBER_POSITION' value becomes invalid or undefined, then this column field should be filled with the FITS null value appropriate for the column format.

If some or all of the group members reside in FITS files separate from the group table itself then the following two columns are also required:

- `TTYPEn␣␣=␣'MEMBER_LOCATION'` – **character field:** Contains the location of the group member's FITS file using Uniform Resource Identifiers. If the FITS file resides on the same computer system as the group table, then partial URIs may be used instead of absolute URIs. If the group member resides in the same FITS file as the group table, or the MEMBER_LOCATION value becomes invalid then this field may be filled with the FITS null value appropriate for the column type.

- `TTYPEn␣␣=␣'MEMBER_URI_TYPE'` – **character field:** Contains the mnemonic for the Uniform Resource Identifier type used in the corresponding MEMBER_LOCATION field. Recommended values for this column field are 'URL' for the Uniform Resource Locator and 'URN' for the Uniform Resource Name. As other URI types are defined their mnemonics will also become acceptable values for this field. In cases where the MEMBER_URI_TYPE is undefined (such as a null or blank MEMBER_LOCATION field value) this field may contain the FITS null value appropriate for the column type.

Besides the table columns defined above, a group table may contain any number of user defined columns. Group table columns may appear in any order within the table and their TTYPEn values are not to be considered case-sensitive.

3 Keywords for Group Member Extensions

No additional keywords are required for HDUs that are members of a group. This rule is to ensure that all currently existing FITS files and their constituent HDUs may all be part of this convention. There are, however, several grouping related keywords whose presence is strongly recommended in newly created headers. The description of these keywords follow.

- **EXTNAME (character):** This keyword is the FITS reserved keyword EXTNAME. The use of EXTNAME allows HDUs of a given XTENSION type with similar structure and content to be identified with a common name tag. Additionally, the grouping convention uses EXTNAME to identify group members by reference (see section 1). For any HDU belonging to a group, the combination of XTENSION, EXTNAME and EXTVER keyword values should uniquely identify the HDU within its FITS file. An exception to this rule occurs when group tables are themselves members of a group. In this case the combination of EXTNAME and EXTVER keyword values alone must uniquely identify the HDU within its FITS file. This is because within a given FITS file the group tables may be built from a mix of ASCII (`XTENSION=␣'TABLE␣␣␣'`) and binary tables (`XTENSION=␣'BINTABLE'`).

- **EXTVER (integer):** This keyword is the FITS reserved keyword EXTVER. The use of EXTVER allows unique identification of HDUs with a given XTENSION type and EXTNAME value. Additionally, the grouping convention uses EXTVER to identify group members by reference (see section 1). For any HDU belonging to a group, the combination of XTENSION, EXTNAME and EXTVER keyword values should uniquely identify the HDU within its FITS file; however, please note the exception outlined above.

- **GRPIDn (integer):** A series of indexed keywords, where 'n' is a positive integer index number starting with the value 1 for the first keyword, and increasing by 1 for each subsequent keyword, that denote the group(s) to which an HDU belongs. The value of GRPIDn is the EXTVER value of the nth group table that the HDU is a member of. In this sense, the EXTVER value of a group table defines a unique ID for the group within a FITS file. If the value of GRPIDn is negative, then the HDU is a member of a group defined in another file. In this case the absolute value of GRPIDn is the EXTVER value of the external group table, and the corresponding GRPLCn keyword holds the URI of the FITS file containing the group's table. The GRPIDn keywords (and their associated GRPLCn keywords) not only identify HDUs as members of groups, but also allow group members to "point" back to their group tables. Any software that might change the position or nature of the HDU would know that it was a member of a group and that the group table would require updating. The GRPIDn and GRPLCn keywords are only used to point back to the grouping tables that directly include that HDU as a member. These keywords should not be used to point back to higher level grouping tables, if a member's grouping table is itself a member of a higher-level group.

- **GRPLCn (character):** A series of indexed keywords, where 'n' is a positive integer index number starting with the value 1 for the first keyword, and increasing by 1 for each subsequent keyword, that contain the Uniform Resource Identifiers corresponding to the GRPIDn keyword. The GRPLCn values follow the same syntax rules as those specified for the group table's MEMBER_LOCATION column (see section 2.2). It is unnecessary to have a GRPLCn keyword accompany a GRPIDn keyword when the value of the GRPIDn keyword is positive. Alternatively, the value of the GRPLCn keywords may be reference strings that refer to the member's group table HDU (see section 5).

4 Example Group Table Headers

The following are examples of valid group table headers that use different combinations of identification methods.

Example 1: A group containing five members all of which reside in the same file as the group table. This group is itself a member of two other groups and both of those groups' tables reside in the same file as this extension. The member position identification method is used to locate member HDUs.

```
XTENSION= 'BINTABLE'        / This is a binary table
BITPIX  =                 8 / Table contains 8-bit bytes
NAXIS   =                 2 / Number of axis
NAXIS1  =                 4 / Width of table in bytes
NAXIS2  =                 5 / Number of member entries
GCOUNT  =                 1 / Mandatory FITS keyword
PCOUNT  =                 0 / Number of bytes in HEAP area
TFIELDS =                 1 / Number of columns in table
EXTNAME = 'GROUPING'        / This BINTABLE contains a group
EXTVER  =                 3 / The ID number of this group
GRPID1  =                 1 / Part of group 1
GRPID2  =                 2 / Part of group 2
TTYPE1  = 'MEMBER_POSITION' / Position of member within file
TFORM1  = '1J'              / Datatype descriptor
END
```

Example 2: A group containing 150 members, some of which reside in FITS files different from that of the group table. All member identification methods are used.

```
XTENSION= 'BINTABLE'        / This is a binary table
BITPIX  =                 8 / Table contains 8-bit bytes
NAXIS   =                 2 / Number of axis
NAXIS1  =                79 / Width of table in bytes
NAXIS2  =               150 / Number of member entries
GCOUNT  =                 1 / Mandatory FITS keyword
PCOUNT  =                 0 / Number of bytes in HEAP area
TFIELDS =                 6 / Number of columns in table
EXTNAME = 'GROUPING'        / This BINTABLE contains a group
EXTVER  =                 7 / The ID number of this group
TTYPE1  = 'MEMBER_LOCATION' / URI of file containing member HDU
TFORM1  = '30A     '        / Datatype descriptor
TTYPE2  = 'MEMBER_URI_TYPE' / URI type of MEMBER_LOCATION field
TFORM2  = '3A      '        / Datatype descriptor
TTYPE3  = 'MEMBER_POSITION' / Position of member within file
TFORM3  = '1J      '        / Datatype descriptor
TTYPE4  = 'MEMBER_XTENSION' / XTENSION keyword value of member
TFORM4  = '8A      '        / Datatype descriptor
TTYPE5  = 'MEMBER_NAME'     / EXTNAME keyword value of member
TFORM5  = '30A     '        / Datatype descriptor
TTYPE6  = 'MEMBER_VERSION'  / EXTVER keyword value of member
TFORM6  = '1J      '        / Datatype descriptor
END
```

Location	URI Type	Position	XTENSION	EXTNAME	EXTVER
/data/file1.fits	URL	2	null	null	null
/data/file2.fits	URL	null	BINTABLE	EVENTS	1
null	null	0	null	null	null
null	null	null	IMAGE	SKY	1
...	...	...	...	...	...

In this example, the first member HDU is located in the 2nd extension of the file /data/file1.fits. The 2nd member HDU is located in the binary table extension in the file /data/file2.fits that has EXTNAME = 'EVENTS' and EXTVER = 1. The 3rd and 4th member HDUs are located in the same file as this grouping table; the 3rd member is the primary array (since Position = 0), and the 4th member is the image extension that has EXTNAME = 'SKY' and EXTVER = 1.

Example 3: A group containing 17 members, some of which reside in FITS files different from that of the group table. This group is a member of six other groups, two of which are defined in FITS files on other computer systems and one that is defined in a FITS file on the same computer system. The member reference identification and member file location methods are used. Two user defined columns are also present.

```
XTENSION= 'BINTABLE'         / This is a binary table
BITPIX  =                  8 / Table contains 8-bit bytes
NAXIS   =                  2 / Number of axis
NAXIS1  =                180 / Width of table in bytes
NAXIS2  =                 17 / Number of member entries
GCOUNT  =                  1 / Mandatory FITS keyword
PCOUNT  =                  0 / Number of bytes in HEAP area
TFIELDS =                  7 / Number of columns in table
EXTNAME = 'GROUPING'         / This BINTABLE contains a group
EXTVER  =                  7 / The ID number of this group
GRPID1  =                  3 / Member of group 3
GRPID2  =                  6 / Member of group 6
GRPID3  =                 18 / Member of group 18
GRPID4  =                 -1 / Member of external group 1
GRPLC4  = 'http://fits.gsfc.nasa.gov/FITS/file1.fits' / location of
COMMENT                         FITS file containing group
GRPID5  =                 -5 / Member of external group 5
GRPLC5  = '/FITS/file5.fits' / Location of file containing group
GRPID6  =                 -2 / Member of external group 2
GRPLC6  = 'http://www.noao.edu/irafdir/file2.fits' / location of
COMMENT                         FITS file containing group
TTYPE1  = 'USER_INFO_1'      / A user supplied column
TFORM1  = '25J     '         / Datatype descriptor
TTYPE2  = 'MEMBER_LOCATION'  / URI of file containing member HDU
TFORM2  = '30A     '         / Datatype descriptor
TTYPE3  = 'MEMBER_XTENSION'  / XTENSION keyword value of member
TFORM3  = '8A      '         / Datatype descriptor
TTYPE4  = 'MEMBER_NAME'      / EXTNAME keyword value of member
TFORM4  = '30A     '         / Datatype descriptor
TTYPE5  = 'USER_INFO_2'      / A user supplied column
TFORM5  = '5A      '         / Datatype descriptor
TTYPE6  = 'MEMBER_VERSION'   / EXTVER keyword value of member
TFORM6  = '1J      '         / Datatype descriptor
TTYPE7  = 'MEMBER_URI_TYPE'  / URI type of MEMBER_LOCATION field
TFORM7  = '3A      '         / Datatype descriptor
END
```

Example 4: A group containing 82 members, some of which reside in FITS files different from that of the group table. This group is a member of three other groups, and makes use of the member position and member file location methods. One user defined column is present. Note that in this example an ASCII table (as opposed to a binary table) is used to define the group.

```
XTENSION= 'TABLE   '         / This is an ASCII table
BITPIX  =                  8 / Table contains 8-bit ASCII characters
NAXIS   =                  2 / Number of axis
NAXIS1  =                 46 / Width of table in bytes
NAXIS2  =                 82 / Number of member entries
GCOUNT  =                  1 / Mandatory FITS keyword
PCOUNT  =                  0 / Mandatory FITS keyword
TFIELDS =                  4 / Number of columns in table
EXTNAME = 'GROUPING'         / This TABLE contains a group
EXTVER  =                 31 / The ID number of this group
GRPID1  =                  3 / Member of group 3
GRPID2  =                  9 / Member of group 9
GRPID3  =                 27 / Member of group 27
TTYPE1  = 'USER_INFO_1'      / A user supplied column
TFORM1  = 'E10.3   '         / Datatype descriptor
TBCOL1  =                  1 / Starting table column for field
TTYPE2  = 'MEMBER_LOCATION'  / URI of file containing member HDU
TFORM2  = 'A30     '         / Datatype descriptor
TBCOL2  =                 11 / Starting table column for field
TTYPE3  = 'MEMBER_URI_TYPE'  / URI type of MEMBER_LOCATION field
TFORM3  = 'A3      '         / Datatype descriptor
TBCOL3  =                 41 / Starting table column for field
TTYPE4  = 'MEMBER_POSITION'  / XTENSION keyword value of member
TFORM4  = 'I3      '         / Datatype descriptor
TBCOL4  =                 44 / Starting table column for field
END
```

5 Acknowledgments

We gratefully acknowledge the support of the NASA Applied Information Systems Research Program, underwhich this effort is partially funded.

6 Appendix I. Reference Strings

This appendix is provided for informational purposes and is not directly necessary to the grouping convention itself. It defines a syntax by which any keyword value or column entry can point to the location of another HDU.

In certain circumstances, it may be convenient to point, or *refer*, to a HDU from another HDU. Such references neither imply or require the hierarchical association information as allowed by grouping table structures, but still serve a similar function by pointing to another data structure residing in a separate HDU.

If *referring* to a single HDU is preferable to forming a hierarchical association and including the given HDU as a member, then keyword and table column values may employ the same syntax as used for the identification of group members. For notational convenience, thus allowing all the information to be included in a single keyword value or table column entry, the reference should be expressed as a single character string of either type 1 format,

'MEMBER_LOCATION':'MEMBER_XTENSION':'MEMBER_EXTNAME':'MEMBER_EXTVER'

or of type 2 format,

'MEMBER_LOCATION':'MEMBER_POSITION'

where each quantity enclosed in single quotation marks is replaced by its corresponding value as defined in section 2.2. The colons (':', ASCII 58) appearing in the expressions are significant and must be used to separate the fields of the string. Such expressions are known as *reference strings*.

Default values in the HDU reference strings are permitted but must obey the following rules. Note that by implication a reference string may begin with a colon field separator (':', ASCII 58) but may not terminate with a colon field separator.

- For type 1 format reference strings, the 'MEMBER_XTENSION' and 'MEMBER_EXTNAME' fields must always be specified.
- For type 1 format reference strings, a non-existent 'MEMBER_EXTVER' is permitted and infers an EXTVER value of 1.
- For type 2 format reference strings, one of the two possible fields ('MEMBER_LOCATION' or 'MEMBER_POSITION') must always be specified, but see the rule below on non-existent 'MEMBER_LOCATION' fields.
- Type 1 and type 2 format reference strings with a non-existent 'MEMBER_LOCATION' value are permitted and infer that the referred-to HDU resides in the same FITS file as the HDU containing the reference string. To denote the absence of the 'MEMBER_LOCATION' value, the first character of the reference string shall be a colon (':', ASCII 58).
- A reference string containing only the 'MEMBER_LOCATION' field shall infer a type 2 format with a 'MEMBER_POSITION' value of 1 (i.e., the first non-primary array FITS file extension). Note that a reference string of this form completely conforms to the syntax of a URI.

Below are examples of valid reference strings. In each case the following values are assumed:

- 'MEMBER_LOCATION' = file://www.archive.edu/archive/sample.fits,
- 'MEMBER_XTENSION' = BINTABLE,
- 'MEMBER_EXTNAME' = EVENTS,
- 'MEMBER_EXTVER' = 1, and
- 'MEMBER_POSITION' = 1.

Note that the values of 'MEMBER_EXTVER' and 'MEMBER_POSITION' chosen for the examples demonstrate the use of the default reference string fields; the choice of different values would make the default cases invalid.

- If the referenced HDU resides in a different FITS file and on a different computer system:
  - file://www.archive.edu/archive/sample.fits:BINTABLE:EVENTS:1
  - file://www.archive.edu/archive/sample.fits:BINTABLE:EVENTS
    (note: using default value for 'MEMBER_EXTVER')
  - file://www.archive.edu/archive/sample.fits:1
  - file://www.archive.edu/archive/sample.fits
    (note: using default value for 'MEMBER_POSITION')
- If the referenced HDU resides in a different FITS file but on the same computer system:
  - /archive/sample.fits:BINTABLE:EVENTS:1
    (note: absolute file path specified)
  - archive/sample.fits:BINTABLE:EVENTS:1
    (note: relative file path specified)
  - sample.fits:BINTABLE:EVENTS:1
    (note: relative file path specified)
  - /archive/sample.fits:BINTABLE:EVENTS
    (note: using default value for 'MEMBER_EXTVER')
  - /archive/sample.fits:1
  - sample.fits
    (note: using default value for 'MEMBER_POSITION')
- If the referenced HDU resides in the same FITS file:
  - :BINTABLE:EVENTS:1
  - :BINTABLE:EVENTS
    (note: using default value for 'MEMBER_EXTVER')
  - :1

Please note that reference strings are meant only to supplement and enhance the hierarchical grouping convention as described above. In particular, reference strings should be used sparingly and with care; they do not provide the same level of data format structure and long-term archival stability as the grouping tables themselves.

7 Appendix II. Application Program Interface

This appendix describes an application program interface (API) in ANSI C that was implimented by the ISDC to facility creating and managing grouping tables by the INTEGRAL mission application software. Use of this particular API is not required and is shown here only for informational purposes. This API software is distributed and supported as a component of the CFITSIO software library which is maintained by the HEASARC at NASA/GSFC.

This API provides functions for the creation and manipulation of FITS HDU Groups, as defined in the "Hierarchical Grouping Convention for FITS". A group is a collection of HDUs whose association is defined by a *grouping table*. HDUs which are part of a group are referred to as *member HDUs* or simply as *members*. Grouping table member HDUs may themselves be grouping tables, thus allowing for the construction of open-ended hierarchies of HDUs.

Grouping tables contain one row for each member HDU. The grouping table columns provide identification information that allows applications to reference or "point to" the member HDUs. Member HDUs are expected, but not required, to contain a set of GRPIDn/GRPLCn keywords in their headers for each grouping table that they are referenced by. In this sense, the GRPIDn/GRPLCn keywords "link" the member HDU back to its Grouping table. Note that a member HDU need not reside in the same FITS file as its grouping table. A given HDU may belong to any number of groups, however the HDU is limited to a maximum of 999 GRPIDn/GRPLCn keywords that point back to the grouping tables to which it is a member.

Grouping tables are implemented as FITS binary tables with up to six pre-defined column TTYPEn values:
'MEMBER_XTENSION', 'MEMBER_NAME',
'MEMBER_VERSION', 'MEMBER_POSITION',
'MEMBER_URI_TYPE' and 'MEMBER_LOCATION'.
The first three columns allow member HDUs to be identified by reference to their XTENSION, EXTNAME and EXTVER keyword values. The fourth column allows member HDUs to be identified by HDU position within their FITS file. The last two columns identify the FITS file in which the member HDU resides, if different from the grouping table FITS file.

Additional user defined "auxiliary" columns may also be included with any grouping table. When a grouping table is copied or modified the presence of auxiliary columns is always taken into account by the grouping support functions; however, the grouping support functions cannot directly make use of this data.

If a grouping table column is defined but the corresponding member HDU information is unavailable then a null value of the appropriate data type is inserted in the column field. Integer columns (MEMBER_POSITION, MEMBER_VERSION) are defined with a TNULLn value of zero (0). Character field columns (MEMBER_XTENSION, MEMBER_NAME, MEMBER_URI_TYPE, MEMBER_LOCATION) utilize an ASCII null character to denote a null field value.

The grouping support functions belong to two basic categories: those that work with grouping table HDUs and those that work with member HDUs. Two functions, fits_copy_group() and fits_remove_group(), have the option to recursively copy/delete entire groups. Care should be taken when employing these functions in recursive mode as poorly defined groups could cause unpredictable results. The problem of a grouping table directly or indirectly referencing itself (thus creating an infinite loop) is protected against; in fact, neither function will attempt to copy or delete an HDU twice.

7.1 Grouping Table Routines

1. Create (append) a grouping table at the end of the current FITS file pointed to by fptr. The grpname parameter provides the grouping table name (GRPNAME keyword value) and may be set to NULL if no group name is to be specified. The grouptype parameter specifies the desired structure of the grouping table and may take on the values: GT_ID_ALL_URI (all columns created), GT_ID_REF (ID by reference columns), GT_ID_POS (ID by position columns), GT_ID_ALL (ID by reference and position columns), GT_ID_REF_URI (ID by reference and FITS file URI columns), and GT_ID_POS_URI (ID by position and FITS file URI columns).

```
int fits_create_group(fitsfile *fptr, char *grpname,
                      int grouptype, int *status)
```

2. Create (insert) a grouping table just after the CHDU of the current FITS file pointed to by fptr. All HDUs below the the insertion point will be shifted downwards to make room for the new HDU. The grpname parameter provides the grouping table name (GRPNAME keyword value) and may be set to NULL if no group name is to be specified. The grouptype parameter specifies the desired structure of the grouping table and may take on the values: GT_ID_ALL_URI (all columns created), GT_ID_REF (ID by reference columns), GT_ID_POS (ID by position columns), GT_ID_ALL (ID by reference and position columns), GT_ID_REF_URI (ID by reference and FITS file URI columns), and GT_ID_POS_URI (ID by position and FITS file URI columns).

```
int fits_insert_group(fitsfile *fptr, char *grpname,
                      int grouptype, int *status)
```

3. Change the structure of an existing grouping table pointed to by gfptr. The grouptype parameter (see fits_create_group() for valid parameter values) specifies the new structure of the grouping table. This function only adds or removes grouping table columns, it does not add or delete group members (i.e., table rows). If the grouping table already has the desired structure then no operations are performed and function simply returns with a (0) success status code. If the requested structure change creates new grouping table columns, then the column values for all existing members will be filled with the null values appropriate to the column type.

```
int fits_change_group(fitsfile *gfptr, int grouptype, int *status)
```

4. Remove the group defined by the grouping table pointed to by gfptr, and optionally all the group member HDUs. The rmopt parameter specifies the action to be taken for all members of the group defined by the grouping table. Valid values are: OPT_RM_GPT (delete only the grouping table) and OPT_RM_ALL (recursively delete all HDUs that belong to the group). Any groups containing the grouping table gfptr as a member are updated, and if rmopt == OPT_RM_GPT all members have their GRPIDn and GRPLCn keywords updated accordingly. If rmopt == OPT_RM_ALL, then other groups that contain the deleted members of gfptr are updated to reflect the deletion accordingly.

```
int fits_remove_group(fitsfile *gfptr, int rmopt, int *status)
```

5. Copy (append) the group defined by the grouping table pointed to by infptr, and optionally all group member HDUs, to the FITS file pointed to by outfptr. The cpopt parameter specifies the action to be taken for all members of the group infptr. Valid values are: OPT_GCP_GPT (copy only the grouping table) and OPT_GCP_ALL (recursively copy ALL the HDUs that belong to the group defined by infptr). If the cpopt == OPT_GCP_GPT then the members of infptr have their GRPIDn and GRPLCn keywords updated to reflect the existence of the new grouping table outfptr, since they now belong to the new group. If cpopt == OPT_GCP_ALL then the new grouping table outfptr only contains pointers to the copied member HDUs and not the original member HDUs of infptr. Note that, when cpopt == OPT_GCP_ALL, all members of the group defined by infptr will be copied to a single FITS file pointed to by outfptr regardless of their file distribution in the original group.

```
int fits_copy_group(fitsfile *infptr, fitsfile *outfptr,
                    int cpopt, int *status)
```

6. Merge the two groups defined by the grouping table HDUs infptr and outfptr by combining their members into a single grouping table. All member HDUs (rows) are copied from infptr to outfptr. If mgopt == OPT_MRG_COPY then infptr continues to exist unaltered after the merge. If the mgopt == OPT_MRG_MOV then infptr is deleted after the merge. In both cases, the GRPIDn and GRPLCn keywords of the member HDUs are updated accordingly.

```
int fits_merge_groups(fitsfile *infptr, fitsfile *outfptr,
                      int mgopt, int *status)
```

7. "Compact" the group defined by grouping table pointed to by gfptr. The compaction is achieved by merging (via fits_merge_groups()) all direct member HDUs of gfptr that are themselves grouping tables. The cmopt parameter defines whether the merged grouping table HDUs remain after merging (cmopt == OPT_CMT_MBR) or if they are deleted after merging (cmopt == OPT_CMT_MBR_DEL). If the grouping table contains no direct member HDUs that are themselves grouping tables then this function does nothing. Note that this function is not recursive, i.e., only the direct member HDUs of gfptr are considered for merging.

```
int fits_compact_group(fitsfile *gfptr, int cmopt, int *status)
```

8. Verify the integrity of the grouping table pointed to by gfptr to make sure that all group members are accessible and that all links to other grouping tables are valid. The firstfailed parameter returns the member ID (row number) of the first member HDU to fail verification (if positive value) or the first group link to fail (if negative value). If gfptr is successfully verified then firstfailed contains a return value of 0.

```
int fits_verify_group(fitsfile *gfptr, long *firstfailed, int *status)
```

9. Open a grouping table that contains the member HDU pointed to by mfptr. The grouping table to open is defined by the grpid parameter, which contains the keyword index value of the GRPIDn/GRPLCn keyword(s) that link the member HDU mfptr to the grouping table. If the grouping table resides in a file other than the member HDUs file then an attempt is first made to open the file readwrite, and failing that readonly. A pointer to the opened grouping table HDU is returned in gfptr.

   Note that it is possible, although unlikely and undesirable, for the GRPIDn/GRPLCn keywords in a member HDU header to be non-continuous, e.g., GRPID1, GRPID2, GRPID5, GRPID6. In such cases, the grpid index value specified in the function call shall identify the (grpid)th GRPID value. In the above example, if grpid == 3, then the group specified by GRPID5 would be opened.

```
int fits_open_group(fitsfile *mfptr, int group,
                    fitsfile **gfptr, int *status)
```

10. Add a member HDU to an existing grouping table pointed to by gfptr. The member HDU may either be pointed to mfptr (which must be positioned to the member HDU) or, if mfptr == NULL, identified by the hdupos parameter (the HDU position number, Primary array == 1) if both the grouping table and the member HDU reside in the same FITS file. The new member HDU shall have the appropriate GRPIDn and GRPLCn keywords created in its header. Note that if the member HDU is already a member of the group then it will not be added a second time.

```
int fits_add_group_member(fitsfile *gfptr, fitsfile *mfptr,
                          int hdupos, int *status)
```

7.2 Group Member Routines

1. Return the number of member HDUs in a grouping table gfptr. The number member HDUs is just the NAXIS2 value (number of rows) of the grouping table.

```
int fits_get_num_members(fitsfile *gfptr, long *nmembers,
                         int *status)
```

2. Return the number of groups to which the HDU pointed to by mfptr is linked, as defined by the number of GRPIDn/GRPLCn keyword records that appear in its header. Note that each time this function is called, the indices of the GRPIDn/GRPLCn keywords are checked to make sure they are continuous (ie no gaps) and are re-enumerated to eliminate gaps if found.

```
int fits_get_num_groups(fitsfile *mfptr, long *nmembers,
                        int *status)
```

3. Open a member of the grouping table pointed to by gfptr. The member to open is identified by its row number within the grouping table as given by the parameter 'member' (first member == 1) . A fitsfile pointer to the opened member HDU is returned as mfptr. Note that if the member HDU resides in a FITS file different from the grouping table HDU then the member file is first opened readwrite and, failing this, opened readonly.

```
int fits_open_member(fitsfile *gfptr, long member,
                     fitsfile **mfptr, int *status)
```

4. Copy (append) a member HDU of the grouping table pointed to by gfptr. The member HDU is identified by its row number within the grouping table as given by the parameter 'member' (first member == 1). The copy of the group member HDU will be appended to the FITS file pointed to by mfptr, and upon return mfptr shall point to the copied member HDU. The cpopt parameter may take on the following values: OPT_MCP_ADD which adds a new entry in gfptr for the copied member HDU, OPT_MCP_NADD which does not add an entry in gfptr for the copied member, and OPT_MCP_REPL which replaces the original member entry with the copied member entry.

```
int fits_copy_member(fitsfile *gfptr, fitsfile *mfptr,
                     long member, int cpopt, > int *status)
```

5. Transfer a group member HDU from the grouping table pointed to by infptr to the grouping table pointed to by outfptr. The member HDU to transfer is identified by its row number within infptr as specified by the parameter 'member' (first member == 1). If tfopt == OPT_MCP_ADD then the member HDU is made a member of outfptr and remains a member of infptr. If tfopt == OPT_MCP_MOV then the member HDU is deleted from infptr after the transfer to outfptr.

```
int fits_transfer_member(fitsfile *infptr, fitsfile *outfptr,
                         long member, int tfopt, int *status)
```

6. Remove a member HDU from the grouping table pointed to by gfptr. The member HDU to be deleted is identified by its row number in the grouping table as specified by the parameter 'member' (first member == 1). The rmopt parameter may take on the following values: OPT_RM_ENTRY which removes the member HDU entry from the grouping table and updates the member's GRPIDn/GRPLCn keywords, and OPT_RM_MBR which removes the member HDU entry from the grouping table and deletes the member HDU itself.

```
int fits_remove_member(fitsfile *fptr, long member,
                       int rmopt, int *status)
```

8 References

Berners-Lee, Tim, 1994. "World Wide Web Initiative", CERN - European Particle Physics Lab. http://info.cern.ch/hypertext/WWW /TheProject.html .

Cotton, W. D., Tody, D. and Pence W., 1994. "Binary Table Extension to FITS: A Proposal", version dated June 13, 1994.

Grosbøl, P., Harten, R. H., Greisen, E. W., and Wells, D. C., 1988. "Generalized extensions and blocking factors for FITS," Astronomy and Astrophysics Suppl., 73, 359-364.

Harten, R. H., Grosbøl. P., Greisen, E. W., and Wells, D. C., 1988. "The FITS tables extension", Astronomy and Astrophysics Suppl., 73, 365-372.

Ponz, J. D., Thompson, R. W., and Munoz, J. R., 1994. "FITS Image Extension" , Astronomy and Astrophysics Suppl., vol 105, pp 53-55.