# dsslib

June 2, 2019

**Abstract**

Library routines for accessing and manipulating data subspace specifications.

## 1 Description

The data subspace stores in a FITSfile the filtering criteria used to select the data in that file. In our case what we are really talking about is storing in an event list file the selection criteria used to select the events in that file. This information will also be stored in data products produced from these event lists, but the DSS specification essentially refers back to the event list used to create the products, and it is probably easiest to think of it in terms of an event list.

We can think of each of the columns in an event list as an axes in a multi-dimensional data space. Thus each event (row) represents a point in this space. When you filter an event list, what you are doing is specifying one or more volumes in the data space into which data points must lie. This is the data subspace (DSS) for this filtered event list. So what one must specify in the data subspace is what axes (columns) are filtered on, and what restrictions were applied to each of those axes.

In the nomenclature used in the API, a data subspace is made of one or more components, each component specify a volume in data space. Each component is made up of one or more filters, each filter specifying a data axis, and the filtering criteria for this axis. Another way to think of this is that each filter represents a boolean expression for a particular value. Then a data component is a set of logically ANDed filters, and a data subspace is a set of logically ORed components. It is assumed that if a particular axis does not have a filter specified, then all values for that axis are allowed.

Several types of data filters are supported. The simplest is a range filter, which specifies a list of valid ranges for a data value. Bit mask filters are supported, which conceptually are just a special case of the range filter. The special case of GTI filters are also supported.

We also support two-dimensional filters, which specify a geometrical region. In this case it is necessary to specify those data axes which represent the $X$ and $Y$ axes for the region. These axes are called the components of the 2-d filter (note difference in usage of the word component here versus above). The 2-d system itself is also considered to be a DSS axis.

The most common use of 2-d filters is for coordinate systems (`X-Y` , `DETX-DETY` , etc). Special conventions are detailed in [1], whereby a certain 2-d axis name implies it has certain component names. For example, if the axis name is `POS`, then by convention it is a $2D$ axis with components `X` and `Y`. I reproduce below the table from [1] detailing the conventions:

| Component names | Default composite name |
|---|---|
| `X,Y` | `POS` |
| `*X,*Y` | `*`, for all `*` |
| `RA,DEC` | `EQPOS` |
| `GLON,GLAT` | `GALPOS` |

## 1.1 Data Subspace Expression Parser

Tools are provided for converting a restricted set of **selectlib** expressions into a data subspace specification (see the **selectlib** documentation for details of the expression format). A parser reads the selection expression and generates subspace filters and components as appropriate. However, not all selection expressions can be expressed as data subspace specifications. Those types of expressions which cannnot be handled by the parser involve arithmetic operations on column values. Arithmetic expressions of constants are allowed, however.

$$RAWX + 1 > 50 \qquad \text{NOT allowed}$$

$$RAWY > 50{*}{*}2 \qquad \text{allowed}$$

### 1.1.1 Tips and Grouping of Region Functions

The data subspace in some cases will produce more compact subspace specifications than in others. For range filters, it is always better to use the "in" operator than to specify ranges using >, <, etc. Though the two examples below are mathematically equivalent, the second example results in a more compact subspace specification than the first.

$$RAWX < 50 \;\&\&\; RAWX > 25 \qquad \text{allowed, but discouraged}$$

$$RAWX \; in \; (25 : 50) \qquad \text{allowed}$$

There is also a special subtlety with regards to the grouping of region function filters. When the parser sees the expression below,

$$(A, B) \; IN \; ellipse(100, 200, 30, 40, 0) \;\&\&\; (A, B) \; in \; polygon(100, 200, 1, 1) \qquad \text{allowed}$$

it tries to combine these into one filter with two regions. If the cooridate specifications do not match, then it returns an error.

$$(A, B) \; IN \; ellipse(100, 200, 30, 40, 0) \;\&\&\; (A, BX) \; in \; polygon(100, 200, 1, 1) \qquad \text{NOT allowed}$$

If, however, the region functions are isolated in parentheses, then the parser will put them in seperate filters. Thus the following expression is allowed.

$((A, B)\ IN\ ellipse(100, 200, 30, 40, 0))\ \&\&\ ((A, BX)\ in\ polygon(100, 200, 1, 1))$ allowed

Note that putting in the parentheses unnecessarily will result in a much less compact data subspace. The following expression is equivalent to the first one in this section, but will result in the creation of two subspace filters, instead of one.

$((A, B)\ IN\ ellipse(100, 200, 30, 40, 0))\ \&\&\ ((A, B)\ in\ polygon(100, 200, 1, 1))$ allowed, but discouraged

## 2 Errors

This section documents warnings and errors generated by this task (if any). Note that warnings and errors can also be generated in the SAS infrastructure libraries, in which case they would not be documented here. Refer to the index of all errors and warnings available in the HTML version of the SAS documentation.

**DSSbitType** *(error)*
Column must be an integer for bit mask filters

**BlockSpec** *(error)*
File/block specification is not in a format supported by the data subspace. The deprecated forms using [] or + cannot be used with the data subspace.

**DssRefClash** *(error)*
The filter references different blocks (referring to different block filters) with the same name. The data subspace cannot be written.

**DssBlockClash** *(warning)*
Name clash while writing Data Subspace. Block already exists in output data set.
*corrective action:* The block is renamed.

**NoFilter** *(error)*
Filter not found in data subspace component.

**NoRange** *(error)*
No range information available for this filter type.

**DSextentType** *(error)*
The filter is not of a type for which a determination of the extent is supported (eg. a mask filter)

**DSextentGTI** *(error)*
The argument of the extent command does not correspond to the name of the GTI filter being queried.

**DSextentRange** *(error)*
The argument of the extent command does not correspond to the name of the Range filter being queried.

**DSextentRegion** *(error)*
The argument of the extent command does not correspond to the name of one of the components of the Region filter.

**NoMask** *(error)*
No bit mask information available for this filter type.

**NoRegion** *(error)*
No region information available for this filter type.

**NoBlock** *(error)*
No block information available for this filter type.

**BlockTypeGti** *(error)*
Block type is not compatible with the gti filter type.

**BlockTypeMask** *(error)*
Block type is not compatible with the mask filter type.

**DSSblockType** *(error)*
The filter type for the specified block could not be determined.

**MaskType** *(error)*
Mask array is not of a supported data type.

**DSStype** *(error)*
Incompatible data types for DSS and column specifications.

**BlockTypeRange** *(error)*
Block type is not compatible with the range filter type.

**RangeType** *(error)*
Columns are not of a supported data type for a range specification.

**BlockTypeRegion** *(error)*
Block type is not compatible with the region filter type.

**ColumnClash** *(error)*
Data subspace filter name and column name are not consistent. Cannot write data subspace.

**BadRegionFilter** *(error)*
A data subspace filter with a region specification has been found. This type of filter is not supported.

**BitVol** *(warning)*
Cannot calculate volume for a bit mask filter.
*corrective action:* Zero volume returned for this filter type.

**RangeVol** *(warning)*
Cannot calculate volume of a range specification with only one limit.
*corrective action:* Zero volume is returned for this filter.

**DSStypeTruncate** *(warning)*
Real-valued ranges are used for an integer-valued column.
*corrective action:* Real-valued ranges are truncated to integers.

**DssSyntaxError** *(error)*
The parser was not able to parse the selection expression so as to convert it to a data subspace specification.

**ArithmeticException** *(error)*
An illegal arithmetic operation was specified in the selection expression.

**MaskSpecOffset** *(error)*
A mask filter was specified with a non-zero offset. The data subspace is only able to store masks with a zero offset.

**DssNameMismatch** *(error)*

Name of coordinate axes in region filter do not match that of preceding filter in same component. See section on "Tips and Grouping of Region Functions" to find out how to avoid this error.

**BitMaskTesting** *(error)*

Bit mask filter must be tested against "== 0". This syntax, along with the equivalent syntax "!(mask!==0)", are the only two that are supported.

**DssFitsWrite** *(warning)*

The data subspace indexing has become to large to fit into keywords of the from nDSVALn (keywords can be on more than 8 characters). Thus no filter information can be written to the FITS header for the filter in question.
*corrective action:* No data subspace information is written for the given filter.

## References

[1] J. McDowel. FITS Keyword Conventions in ASC Data Model files. Technical report, ASC, January 1999. SDS-7.2.