MeasurementSet Selection Syntax

S. Bhatnagar
NRAO, Socorro

June 15, 2007

Abstract

This note describes the syntax for the various expressions for selecting data from the Measurement Set. This is the syntax that is implemented in the `MSSelection` module of CASA. The syntax is derived from an earlier note[1] on syntax definition. All expressions consists of a comma or semi-colon separated list of specifications. Except channel selection, the final product of parsing the expressions is a Table Expression Node (TEN) which can be used to construct a selected Measurement Set. Higher level global methods to return a selected MS, given the user supplied expressions are also provided as part of the `MSSelection` module.

Error handling is done via the C++ exception handling mechanism. The exceptions thrown by the `MSSelection` module are of type `MSSelectionError` which is derived from `AipsError` class. The exceptions generated from each individual expression parser are further specializations of the `MSSeletionError` class. Application layer code therefore can have a finer level control on error handling.

1 General Syntax

A MSSelection expression consists of a comma separated list of specifications. Specifications are typically strings or numbers. Strings and numbers can be mixed to form a single list. Elements of the list which can be converted to integers are treated as integer index specification. Elements which do not get parsed as numbers are treated as strings. Where appropriate, strings are matched against names. Depending upon the content of a string,

[1] http://almasw.hq.eso.org/almasw/bin/view/OFFLINE/DataSelection

it can be used as regular expressions or pattern. Where appropriate, physical quantities (numbers with appropriate units) can also be used.

A blank selection expression is interpreted as ”no selection to be applied to the MS”. Hence a blank expression effectively implies ”select all”.

1.1 Number format

Integers can be of any length (in terms of characters) and composed of the characters in the range 0-9. Where appropriate, negative values can be given using the ’-’ character. Floating point numbers can be in the standard format:

- DIGIT.DIGIT : e.g. 10.56
- DIGIT. : e.g. 10.
- .DIGIT : e.g. .56

or in the mantissa-exponent format (e.g. 10.56e-1). If a floating point number is given where only integers are expected (e.g. indexes), the floating point value is truncated to the nearest integer.

1.2 Range specification

Range of numbers (integers or real numbers) can be given in the format N0~N1. Integer ranges are expanded into a list of integers starting from N0 (inclusive) to N1 (inclusive). Range of real numbers is used to select all values between N0 and N1 (including the boundaries). E.g.

Integer ranges:

- 10~30 implies all integers in the range [10,30]
- 10.1~30.5 implies all integers in the range [10,30]

Floating point ranges:

- 10~30 implies all values in the range [10.0, 30.0]
- 10.5~30.7 implies all values in the range [10.5, 30.7]
- 1.05e1 ~ 3.07e1 implies all values in the range [10.5, 30.7]

1.3 Units

Wherever appropriate, units can be optionally specified. Values with units are converted to the units in the Measurement Set (which uses the MKS-system). For ranges, the units are specified only once (at the end) and it applies to both the range boundaries. E.g.

- 1421.07MHz implies 1421.07e6 Hz
- 1421~1500MHz implies all frequencies in the range [1421.0, 1500.0]MHz.

1.4 Strings

String matching can be done in three ways. Any component of a comma separated list that cannot be parsed as number/number range/physical quantity is treated as a regular expression or a literal string. If the string does not contain the characters '*', '', '' or '?', it is treated as a literal string and used for exact matching. If any of the above mentioned characters are part of the string, it is used as a regular expression. As a result, for most cases, the user does not need to supply any special delimiters for literal strings and/or regular expressions. However if it is required that the string be matched exclusively as a regular expression, it can be supplied within a pair of '/' as delimiters. A string enclosed within double quotes ('"') is used exclusively for pattern matching (patterns are a simplified form of regular expressions - used in most UNIX commands for string matching). Patterns are internally converted to equivalent regular expressions before matching. Read elsewhere (e.g. use command "info regex", or visit http://www.regular-expressions.info) for details of regular expression and patterns.

Strings can include any character except the following:

```
','   ';' '"'  '/'  ':' and NEWLINE
```

(these are reserved characters for MSSelection expression syntax). Strings that do not contain any of the characters used to construct regular expressions or patterns are used for exact matches. Although it is highly discouraged to have name in the database containing the above mentioned reserved characters, if one DOES choose to include the reserved characters are part of names etc., those names can only be matched against quoted strings (since regular expression and patterns are super-set of literal strings. I.e. literal string is a valid regular expression also). This leaves the list '"', '*', '?', '' and '' as the list of printable character that cannot be part of a name (i.e., a name containing this character can never be matched in a MSSelection expression). If a need is felt to include these as well, an escape mechanism can be included later (but I would prefer to enforce that

at least these characters not be part of any name!). Following are some examples of strings/regular expressions/patterns:

- The string LBAND will be used as a literal string for exact match. This will match only ”LBAND”.
- The string `*BAND*` will be used as a string pattern for matching. This will match any string which has ”BAND” in it.
- The string `"*BAND*"` will be used as a string pattern for matching. This will also match any string which has the string ”BAND” in it.
- The string /.+BAND.+/ will be used as a regular expression. This will also match any string which as the string ”BAND” in it. (.+ operator has the same meaning as the ’*’ operator of patterns).

1.5 Handling of blanks

In most cases, blanks are treated as white-spaces (i.e., insertion of blanks anywhere in the expression has no effect), except in the case of Field Selection Expressions (see Section 4). Blanks are allowed as part of the field names. Blanks around the delimiting characters (’,’ , ’;’ , ’&’ etc.) are ignored. For field names, blanks after the first valid name character and before the last valid name character are included as part of the name. Hence

- field=`A , B , C` is same as field=`A,B,C`
- field=`A , BB BB , C` is same as field=`A,BB BB,C` (first name is ”A”, second name is ”BB BB” and the third name is ”C”)
- baseline=`1 , 2 , 3 & 4 , 5 , 6 ; 10 ~ 11 & 20 ~ 30` is same as baseline=`1,2,3&4,5,6;10~11&20~30`

2 Time selection

Time selection expression is a comma separated list of time specifications. Time can be specified in the format YYYYY/MM/DD/HH:MM:SS.FF. Fields (i.e., YYYY, MM, DD, HH, MM, SS and FF), starting from left to right, can be omitted. Omitted fields will be replaced by context sensitive defaults as explained below.

T0, T1 and dT in the following description are time specifications.

2.1 Syntax

- time=`T0~T1`

  Selects all time stamps starting from T0 to T1. Fields missing in T0 are replaced by the fields in the time stamp of the first valid selected row in the MS. Fields missing in T1 are replaced by the corresponding fields of T0 (after its defaults are set).

- time=`T0`

  Selects all time stamps that are within an integration time of T0. Integration time is determined from the first valid selected row (more rigorously, an average integration time should be computed). Default settings for the missing fields of T0 are as described in bullet (1) above.

- time=`T0+dT`

  Selects all time stamps starting from T0 and ending with time value given by T0+dT. Defaults of T0 are set as usual described in bullet (1) above. Defaults for dT are set from the time corresponding to MJD=0. I.e. dT is an specification of length of time from nominal ”start of time” (the time stamp of the first valid selected row of the MS).

- time=>`T0`

  Selects all times greater than T0.

- time=<`T0`

  Selects all times less than T0. Default settings for T0 are as above.

3 Antenna/Baseline Selection

This expression is used to perform baseline based selections. Baseline can be specified as a pair of antenna specifications. Since antenna specification can itself be a list of antennas, the expression allows a rich selection syntax which is simple for simple selections.

ANT in the description below is a comma-separated list of antenna specifications. A baseline specification is an single ANT, ANT followed by ampersand or a pair of ANT separated by ampersand. A baseline expression is single baseline specification or a semi-colon separated list of baseline specifications.

3.1 Syntax

An ANT can be given as a single string (literal/pattern/regular expression), single integer ID, a range of integer IDs or a comma separated list of integers. For VLA-specific reasons (see Section 3.2), **only** for antenna specifications, integers are first converted to strings and matched against the antenna names. E.g.

- `VLA:N1` corresponds to antenna named ”VLA:N1”.
- `1,2,3` corresponds to antennas **named** ”1”, ”2” and ”3”.
- `1~3` corresponds to antennas **named** ”1”, ”2” and ”3”.
- `VLA:N*` corresponds to all antennas who’s name starts with ”VLA:N”.

A baseline specification consists of an ANT, an ANT followed by an ampersand or a pair of antenna specifications separated by ampersand. Formally, baseline specification is of the form ANT[&ANT] (where the part in square brackets is optional).

An atomic ANT selects all baselines containing all the antennas in ANT. ANT& selects only baselines between the list of antennas in the antenna specification. ANT1&ANT2 selects baselines between antennas in ANT1 and ANT2 only. E.g.

- baseline=`1,2,3` selects baseline between antennas 1, 2 and 3 and all other antennas.
- baseline=`1,2,3&` selects baseline between antennas 1,2 and 3 only.
- baseline=`1,2,3 & 4,5,6` selects baselines between antennas 1,2,3 and 4,5,6.

Following are more examples of baseline specification using ranges and names:

- baseline=`1~3` same as baseline=1,2,3
- baseline=`1~3&` same as baseline=1,2,3&
- baseline=`1~3 & 4~5` same as baseline=`1,2,3&4,5,6`
- baseline=`VLA:N* & VLA:E*` selects all baselines between antennas with names starting with ”VLA:N” and ”VLA:E”.

The full baseline selection expression is any of the examples shown above or a semi-colon-separated list of baseline specifications. E.g.

- baseline=1~3& ; 4~5&10~15 ; VLA:N* & VLA:E*

with each elements of the semi-colon separated list being interpreted as explained above.

3.2 Integers-as-names VLA naming convention

Antenna naming convention for VLA is such that the antenna names are actually valid integers converted to strings. While we feel that this is indeed a bad idea and it will be best to translate the VLA antenna names to something like VLA1, VLA2 in the CASA VLA filler (or something that does reflect that its a name and not get confused with integer indexes), for now, to accommodate the VLA tradition, the following logic is used:

Just for antenna selection, a user supplied integer (or integer list) is converted to a string and matched against the antenna name. If that fails, the normal logic of using an integer as an integer and matching it with antenna index is done.

So if the antenna with ID 17 is named ”21”, the string

```
"21",VLA22
```

will expand into an antenna ID list of 17,22 (assuming that the antenna named VLA22 has ID 22).

If we conclude that this style of antenna selection is indeed the way we wish to go, users should be aware that the antenna selection will behave differently for telescopes other than VLA. Assuming that antenna with names ”21”,”17, and ”11” have IDs 1,2,3 for VLA, a selection string ”21,17,11” will select antenna with IDs 1,2,3. For other instruments where this is not the naming convention, the same selection string (”21,17,11”) will select antennas with ID 21, 17, and 11.

3.2.1 Note

1. Selection on polarization is not implemented.

4 Field Selection

4.1 Syntax

Field specifications can be literal field names, regular expressions or patterns. Those fields for which the entry in the NAME column of the the FIELD sub-table match the literal field name/regular expression/pattern are selected.

If a field name/regular expression/pattern fails to match any field name, the given name/regular expression/pattern are matched against the field code. If still no field is selected, an exception of type MSSelectionFieldParseError is thrown.

Field specifications can also be give by their integer IDs. IDs can be a single ID or a range of IDs (N0~N1). Field ID selection can also be done using a boolean expression.

For a field specification of the form ”¿ID”, all field IDs greater than ID are selected. Similarly for ”<ID” all field IDs smaller than ID are selected.

The field selection expression is a comma-separated list of field specifications. E.g.

- field=`1,2,3,4` selects field IDs 1,2,3 and 4
- field=`1~4` same as above
- field=`1~4, VIRGO A , 3C*` selects field IDs 1,2,3,4 field named ”VIRGO A” and all fields with names starting with ”3C”.

5 UV Distance Selection

5.1 Syntax

A uv-distance specification is given as a physical quantity (number with units in the format N[UNIT] where UNIT is optional). This is referred to as UVDIST in the description below. Units are optional with the default unit being meter. Units can be specified as ”m”/”M” (for Mega) or ”k”/”K” (for Kilo) followed by ”m”/”M” (for meter) or ”k”/”K” followed by ”l”/”L” (for lambda or kilo-lambda). User supplied values are converted to internal Measurement Set units using the spectral window sensitive reference frequency.

If only a single UVDIST is specified, all rows that exactly matches the given UVDIST are selected. When UVDIST is given as a range in the format N0~N1[UNIT] (where N0 and N1 are valid numbers), all rows corresponding to the uv-distance between N0 and N1 (N0 and N1 included) are selected.

Rows can also be selected via boolean expressions. When specified in the format ”>UVDIST”, all rows with uv-distance greater than the given uv-distance (converted to the appropriate units) are selected. When specified in the format ”<UVDIST”, all rows with uv-distance less than the given uv-distance (converted to the appropriate units) are selected.

To selected rows with uv-distance within an equal range on either side of a given value, UVDIST can be specified as a percentage of the given value in the format Npercent of the given uv-distance in appropriate units are selected.

The full uv-distance selection expression is a comma-separated list of any of the above mentioned uv-distance specifications. E.g.

- uvdist=`100Kl` selects all baselines for which $\sqrt{u^2 + v^2}$ is equal to 100 Kilo-Lambda.
- uvdist=`100Km` selects all baselines for which $\sqrt{u^2 + v^2}$ is equal to 100 Kilo-meter.

- uvdist=`100~200Kl` selects all baselines for which $\sqrt{u^2 + v^2}$ is in the range [100, 200] Kilo-lambda.
- uvdist=>`100Kl` selects all baselines for which $\sqrt{u^2 + v^2}$ is greater than 100 Kilo-lambda.
- uvdist=<`100Kl` selects all baselines for which $\sqrt{u^2 + v^2}$ is less than 100 Kilo-lambda.

6 Frequency Selection

Frequency selection expression consists of two specifications separated by colon (':') in the form:

```
SPWSPEC[:CHANSPEC]
```

where SPW is the spectral window specification and CHANSPEC is the optional frequency specification for selection within the selected spectral windows. When channel specification is omitted, all channels of the selected SPW are selected.

6.1 Spectral Window Specification Syntax

Spectral windows (SPW) specification can be a single ID or a list of spectral window integer IDs, a spectral window name (as a literal string (for exact match)/regular expression/pattern) or a reference frequency value (value with a unit). A single frequency specification is used for exact match with the REF_FREQUENCY column of the SPECTRAL_WINDOW sub-table. A range of frequencies are used to select all SPWs which are within the given range. The allowed units are Hz, KHz, MHz, GHz or THz.

SPWs can also be selected via a boolean expression for integer IDs.

- ">ID" will select all SPWs with ID greater than the specified value.
- "<ID" will select all SPWs with ID lesser than the specified value.
- ">FREQ" will select all SPWs, the reference frequencies of which are greater than the given frequency converted to Hz.
- "<FREQ" will select all SPWs, the reference frequencies of which are greater than the given frequency converted to Hz.

E.g.

- spw=`1,2,3` select spectral window IDs 1,2 and 3
- spw=`1~3` same as above
- spw=`327MHz` selects spectral window with a reference frequency equal to 327.0 MHz
- spw=`327~610MHz` selects all spectral windows with reference frequencies in the range [327.0, 610.0] MHz.
- spw=`>327MHz` selects all spectral windows with reference frequency greater than 327.0 MHz
- spw=`<327MHz` selects all spectral windows with reference frequency less than 327.0 MHz

6.2 Channel Selection Syntax

MSSelection module is currently used to only parse the channel selection expression to produce the START,STOP,STEP tuples. No channel based selection is actually applied to the data. The list of tuples per SPW is passed to the application program layer and the interpretation of the tuples for actual selection depends on the application programmer.

Channel specification (referred to as CHANSPEC in the following description) is a START,STOP,STEP tuple corresponding to the first frequency channel, the last frequency channel and the step size to increment from the first to the last channel. The START and STOP part of the tuple can be range specification for the range [START, STOP] followed by an optional STEP size.

START and STOP part of the tuple can be specified as a single integer or physical quantity or as a range of integers or physical quantities. A single integer is treated as a channel index and the tuple corresponds to [ID,ID,1]. A single physical quantity is matched with the exact frequency value of the channels and tuple of matched channel index is constructed as [ID,ID,1]. A range of integers given as N0~N1 is used to construct a tuple [N0,N1,1]. Similarly for a range of physical quantities.

If a START,STOP part of the tuple specification is followed by "^STEP", the STEP is used as the third value of the tuple. STEP specification is a single integer or physical quantity. E.g. 0~10^2 is converted to a START,STOP,STEP tuple [0,10,2].

A channel selection expression is a semi-colon separated list of channel specifications. E.g.

- chan=`1;2;3` selects channels 1,2 and 3.
- chan=`1~3` selects channels 1,2 and 3.
- chan=`0~10^2` selects channels in the range [0,10] with a step size of 2
- chan=`1421MHz` selects a frequency channel corresponding to the frequency 1421.0 MHz
- chan=`1421~1500MHz` selects all channels in the range [1412.0, 1500.0] MHz
- chan=`1421~1500MHz^10KHz` selects all channels in the range [1421.0, 1500.0] MHz in steps of 10 KHz.
- chan=`1421~1500MHz^10KHz ; 0~10^2 ; 20 ; 30; 40` selects all channels in the range [1421.0, 1500.0] MHz in steps of 10KHz, all channels with indexes in the range [0,10] in steps of 2 channels and channels 10, 20 and 30.

6.3 Frequency Selection Syntax

The specifications for SPW and channels can be combined to form a fully qualified frequency selection expression in the form SPWSPEC[:CHANSPEC] (square brackets indicate that :CHANSPEC is optional). A frequency selection expression is a comma-separated list of SPWSPEC[:CHANSPEC] specifications. Channel selections apply to all SPWs selected by the SPWSPEC on the left of ':'. E.g.

- freq=`LBAND:1421~1500MHz^10KHz , 327MHz:300~400MHz, 0~4:0~10^2 , 5:20;30;40`

  selects

  - all channels corresponding to channels in the range [1421.0, 1500.0] MHz for the SPW named "LBAND", and
  - all channels corresponding to the range [300.0, 400.0]MHz for the SPW with a reference frequency of 327.0 MHz, and
  - Channels in the range [0,10] in steps of 2 for all SPWs with IDs in the range [0,4], and
  - Channels 20, 30 and 40 for SPW 5

6.3.1 Notes

1. For channel specificities of the type `N1~N2:c0;c1;c2;c3MHz` the list of channel selection is applied to all SPWs in the range [N0, N1]. c3 is converted from physical units to channel index by using the channel width from SPW N1. Is it better to use the min. chan. among the selected SPWs?

2. If channel range is out of bounds, it will be brought within bounds per SPW. If the lower limit of a range is greater than the available outer limit, exception is generated. If the lower limit is less than zero, it is set to zero. If upper limit is greater than the available outer limit, it is set to the available outer limit.

   If a single channel specification is greater than the available outer limit, an exception is thrown. If it is less than zero, it is set to zero.

3. For ranges of physical values, it is assumed that the spectral window sub-table's CHAN_FREQ column has ordered list of channel frequencies.

7 Scan/Sub-array Selection

Scan and sub-arrays selections are purely integer ID based selections. The syntax for the specification of both these is therefore identical.

7.1 Syntax

Scan and Sub-array selection specification is single integer (INT) or a range of integers(N0~N1). Scans and sub-arrays can also be selected via boolean expressions of the type ">INT", ">=INT", "<INT" or "<=INT".

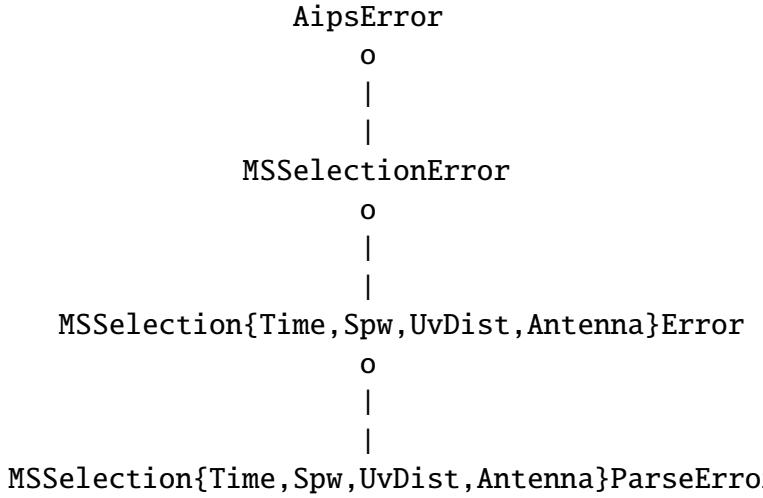
Full expression is a comma-separated list of any of the above index specifications. E.g.

- scan=`0,1,2,3` selects scan numbers 0, 1, 2 and 3
- scan=`0~3` selects scans in the range [0,3]
- scan=`0~3,4,5` selects scans in the range [0,3] and scans 4 and 5
- scan=`>5` selects all scans greater than 5
- scan=`<5` selects all scans less than 5

The getScanList() (getSubArrayList() for sub-arrays) method will return the list generated from INT,INT,... or INT~INT or "<INT" specification. For ">INT" specification, the returned list will be number from INT part of ">INT" upto the maxScans (maxArrays for sub-arrays) set via MSSelection::setMaxScan() (MSSelection::setMaxArray() for sub-arrays). The default value of maxScans/maxArrays is `std::standard_limits<int>::max()` (i.e. the maximum value an integer can take on a machine). It is done this way since there is no efficient way of generating a list of SCAN/ARRAY IDs present in the data and ">INT" is an unbounded list of integers. The list of scans IDs in the data can be **probably** generated using the MSRange class - but I think it can be quite expensive. So if it is necessary to know the precise list of scans selected using the ">INT" specification, the application programs will have to use MSRange class to get the range in the data. An intersection of the list from MSRange and the list from MSSelection.getScanList() will be the actual selected scans (intersection of two CASA vectors can be done using the set_intersection() global method in MSSelectionTools.h).

8 General Error Handling

The MSSelection sub-system reports errors via the C++ exception mechanism. The objects thrown have the following inheritance hierarchy:



All parsing errors are reported by throwing the MSSelection{Time, Spw, Field, UvDist}ParseError exception. All other forms of errors (e.g. illegal range specification N0~N1 where N0 > N1) are reported by throwing an exception of type MSSelectionTime,Spw,Field,UvDistError.

Hence, to catch all errors thrown from the MSSelection sub-system, catch the MSSelectionError object. For more specific exception handling, catch the more qualified MSSelection*Error objects. For catching only parsing errors, catch the MSSelection*ParseError object. As is obvious, any un-caught exception from the MSSelection sub-system will be caught in the AipsError catch block.

The exceptional error message consists of a human understandable one-line description of the error, the string which caused the error and the possible location in the string of the erroneous character. E.g.

Spw Expression: No match found for ”LBAN” (near char. 4 in string ”LBAN”)

9 Examples

Table 1: Summary of the MS used

FIELDID	SPWID	NChan	Pol	NRows	Source Name
0	0	127	RR	10260	0530+135
1	0	127	RR	779139	05582+16320
2	0	127	RR	296190	05309+13319
3	0	127	RR	58266	0319+415
4	0	127	RR	32994	1331+305
5	1	1	RR,RL,LL,RR	23166	KTIP

9.1 Example 1

```
Select field IDs 0,1,2 and field Named KTIP
       Time range: 25/22:40:0 to 26/03:30:0
       Baselines: All baselines between antennas named "1" to "10"
       Spw:  All spectral windows

sbhatnag@atlas>mssplit
ms         = /home/rohir3/sanjay/CASATests/G192.ms/
outms      = tt.ms
field      = 0~2,KTIP
time       = 25/22:40:0 ~ 26/03:30:0
```

```
spw        =
antenna    = 1~10&
uvdist     =
mssplit>go
First selected timestamp = 25-Apr-2003/22:03:42.5
Ant1 = [21, 10, 17, 4, 24, 3, 11, 0, 23, 6]
Ant2 = [21, 10, 17, 4, 24, 3, 11, 0, 23, 6]
Field= [0, 1, 2, 5]
SPW  = []
Number selected rows: 102195
```

9.2 Example 2

```
Select field Named K* (wild card usage)
       Time range: 25/22:40:0 to 26/03:30:0
       Baselines: All baselines between antennas named "1" to "10"
       Spw:  All spectral windows

sbhatnag@atlas>mssplit
ms         = /home/rohir3/sanjay/CASATests/G192.ms/
outms      = tt.ms
field      = K*
time       = 25/22:40:0 ~ 26/03:30:0
spw        =
antenna    = 1~10&
uvdist     =
mssplit>go
First selected timestamp = 26-Apr-2003/03:20:45.0
Ant1 = [21, 10, 17, 4, 24, 3, 11, 0, 23, 6]
Ant2 = [21, 10, 17, 4, 24, 3, 11, 0, 23, 6]
Field= [5]
SPW  = []
Number selected rows: 2376
```

9.3 Example 3

```
-----------------------------------------------------------------------
Select field Named K* (wild card usage)
```

```
       Time range: 25/22:40:0 to 26/03:30:0
       Baselines: All baselines between antennas *named* "1" to "10"
       Spw:  SPW ID 0

       No data should be selected
------------------------------------------------------------------------
sbhatnag@atlas>mssplit
ms         = /home/rohir3/sanjay/CASATests/G192.ms/
outms      = tt.ms
field      = K*
time       = 25/22:40:0 ~ 26/03:30:0
spw        = 0
antenna    = 1~10&
uvdist     =
mssplit>go
First selected timestamp = 25-Apr-2003/22:03:37.5
Ant1 = [21, 10, 17, 4, 24, 3, 11, 0, 23, 6]
Ant2 = [21, 10, 17, 4, 24, 3, 11, 0, 23, 6]
Field= [5]
SPW  = [0]
Number selected rows: 0
```

9.4 Example 4

```
------------------------------------------------------------------------
Select field Named K* (wild card usage)
       Time range: 25/22:40:0 to 26/03:30:0
       Baselines: All baselines between antennas *named* "1" to "10"
       Spw:  SPW ID 1
------------------------------------------------------------------------
sbhatnag@atlas>mssplit
ms         = /home/rohir3/sanjay/CASATests/G192.ms/
outms      = tt.ms
field      = K*
time       = 25/22:40:0 ~ 26/03:30:0
spw        = 1
antenna    = 1~10&
uvdist     =
```

```
mssplit>go
First selected timestamp = 26-Apr-2003/03:20:45.0
Ant1 = [21, 10, 17, 4, 24, 3, 11, 0, 23, 6]
Ant2 = [21, 10, 17, 4, 24, 3, 11, 0, 23, 6]
Field= [5]
SPW  = [1]
Number selected rows: 1188
```

9.5 Example 5

```
------------------------------------------------------------------------
Select field Named K* (wild card usage)
       Time range: 25/22:40:0 to 26/03:30:0
       Baselines: Between antennas "1" to "3" and "4" to "6"
       Spw:  SPW ID 1
------------------------------------------------------------------------
sbhatnag@atlas>mssplit
ms         = /home/rohir3/sanjay/CASATests/G192.ms/
outms      = tt.ms
field      = K*
time       = 25/22:40:0 ~ 26/03:30:0
spw        = 1
antenna    = 1,2,3&4,5,6
uvdist     =
mssplit>go
First selected timestamp = 26-Apr-2003/03:20:45.0
Ant1 = [21, 10, 17]
Ant2 = [4, 24, 3]
Field= [5]
SPW  = [1]
Number selected rows: 198
```

9.6 Example 6

```
------------------------------------------------------------------------
Select field Named K* (wild card usage)
       Time range: 25/22:40:0 to 26/03:30:0
       Baselines: Error in antenna specification
```

```
       Spw:  All spectral windows
--------------------------------------------------------------------
sbhatnag@atlas>mssplit
ms         = /home/rohir3/sanjay/CASATests/G192.ms/
outms      = tt.ms
field      = K*
time       = 25/22:40:0 ~ 26/03:30:0
spw        = 1
antenna    = 1~3 & $~6
uvdist     =
mssplit>go
###AipsError: Antenna Expression: Parse error at or near '%' (near
char. 7 in string "1~3 & $~6")
```

9.7 Example 7

```
--------------------------------------------------------------------
Select field Named K* (wild card usage)
       Time range: 25/22:40:0 to 26/03:30:0
       Baselines: Use antenna ranges
       Spw:  All spectral windows
--------------------------------------------------------------------
sbhatnag@atlas>mssplit
ms         = /home/rohir3/sanjay/CASATests/G192.ms/
outms      = tt.ms
field      = K*
time       = 25/22:40:0 ~ 26/03:30:0
spw        = 1
antenna    = 1~3 & 4~6
uvdist     =
mssplit>go
First selected timestamp = 26-Apr-2003/03:20:45.0
Ant1 = [21, 10, 17]
Ant2 = [4, 24, 3]
Field= [5]
SPW  = [1]
Number selected rows: 198
\end{verbatim>
```

```
\subsection{Example 8}
\begin{verbatim}
-------------------------------------------------------------------
Select field Named K* (wild card usage)
       Time range: 25/22:40:0 to 26/03:30:0
       Baselines: Baselines between antennas "1" to "3" and "4" to "6"
                  and baseline "10"-"11"
       Spw:  All spectral windows
-------------------------------------------------------------------
sbhatnag@atlas>mssplit
ms         = /home/rohir3/sanjay/CASATests/G192.ms/
outms      = tt.ms
field      = K*
time       = 25/22:40:0 ~ 26/03:30:0
spw        = 1
antenna    = 1~3 & 4~6 ; 10&11
uvdist     =
mssplit>go
First selected timestamp = 26-Apr-2003/03:20:45.0
Ant1 = [21, 10, 17, 6]
Ant2 = [4, 24, 3, 1]
Field= [5]
SPW  = [1]
Number selected rows: 231
```

9.8 Example 9

```
-------------------------------------------------------------------
Select field Named K* (wild card usage)
       Time range: Error in range operator
       Baselines: Baselines between antennas "1" to "3" and "4" to "6"
       Spw:  All spectral windows
-------------------------------------------------------------------
sbhatnag@atlas>mssplit
ms         = /home/rohir3/sanjay/CASATests/G192.ms/
outms      = tt.ms
field      = K*
```

```
time       = 25/22:40:0 - 26/03:30:0
spw        = 1
antenna    = 1~3 & 4~6
uvdist     =
mssplit>go
First selected timestamp = 26-Apr-2003/03:20:45.0
###MSSelectionError: MSSelection time error: Parse error at or near
token '-' (near char. 12 in string "25/22:40:0 - 26/03:30:0")
(Did you know we use "~" as the range operator (for a good reason)?)
```

10 Formal Grammar Specifications

10.1 Baseline Selection Expression

```
token IDENTIFIER of type String
token COMMA

token LPAREN
token RPAREN
token WHITE

token INT      of type String
token QSTRING  of type String
token REGEX    of type String

token COLON
token SEMICOLON

type TEN antennastatement
type TEN indexcombexpr
type Vector<int> indexlist
type Vector<int> antidrange
type Vector<int> antidlist
type Vector<int> antid

antennastatement: indexcombexpr
```

```
                | LPAREN indexcombexpr RPAREN

indexcombexpr: indexcombexpr SEMICOLON indexcombexpr
             | indexlist AMPERSAND indexlist
             | indexlist AMPERSAND
             | indexlist

antid: IDENTIFIER
     | QSTRING
     | REGEX

antidrange: INT
          | INT TILDA INT

antidlist: antid
         | antidrange

indexlist: antidlist
         | indexlist COMMA antidlist
         | LPAREN indexlist RPAREN
```

10.2 Frequency Selection Expression

```
token UNIT       of type String
token INT        of type String
token FNUMBER    of type String
token QSTRING    of type String
token REGEX      of type String
token IDENTIFIER of type String

type TEN SpwStatement
type TEN FullSpec
type TEN FullExpr
type Float[2] OneFreq
type Float[2] FreqRange
type Float[2] Physical
type Float[4] IndexRange
```

```
type Float[4] PhyRange
type Vector<int>    Spw
type Float[2] FListElements
type Vector<Float>   FreqList
type Float PhyVal


SpwStatement: FullExpr | LPAREN FullExpr RPAREN

PhyVal: FNUMBER

Physical: PhyVal UNIT

PhyRange: Physical TILDA Physical
        | PhyVal TILDA PhyVal UNIT
        | PhyRange CARET Physical

IndexRange: PhyVal TILDA PhyVal
          | IndexRange CARET PhyVal

FreqRange: IndexRange
         | PhyRange

OneFreq:  PhyVal
        | Physical

FListElements: FreqRange
             | OneFreq

FreqList: FListElements
        | FreqList SEMICOLON FListElements

Spw: IDENTIFIER
   | QSTRING
   | REGEX
   | OneFreq
   | GT OneFreq
   | LT OneFreq
```

```
   | TILDA OneFreq
   | FreqRange

FullSpec: Spw
        | Spw COLON FreqList

FullExpr: FullSpec
        | FullExpr COMMA FullSpec
```

10.3 Field Selection Expression

```
token SQUOTE
token IDENTIFIER  of type String
token COMMA

token LPAREN
token RPAREN
token WHITE

token INT      of type String
token QSTRING  of type String
token REGEX    of type String

token COLON
token SEMICOLON

type TEN fieldstatement
type TEN indexcombexpr
type Vector<int> indexlist
type Vector<int> fieldidrange
type Vector<int> fieldidlist
type Vector<int> fieldid
type Vector<int> fieldidbounds

fieldstatement: indexcombexpr
              | LPAREN indexcombexpr RPAREN
```

```
indexcombexpr: indexlist

fieldid: IDENTIFIER
       | QSTRING
       | REGEX

fielddidrange: INT
            | INT TILDA INT

fielddidbounds: LT INT
             | GT INT
             | GT INT AMPERSAND LT INT

fielddidlist: fieldid
           | fielddidrange
           | fielddidbounds

indexlist : fielddidlist
          | indexlist COMMA fielddidlist
          | LPAREN indexlist RPAREN
```

10.4 Scan/Sub-Array Selection Expression

```
token IDENTIFIER  of type String
token COMMA

token LPAREN
token RPAREN
token WHITE

token INT      of type String
token QSTRING  of type String
token REGEX    of type String

token COLON
token SEMICOLON

type TEN scanstatement
```

```
type TEN indexcombexpr
type TEN scanidrange
type TEN scanidbounds

scanstatement: indexcombexpr
             | LPAREN indexcombexpr RPAREN

indexcombexpr: scanidrange
             | scanidbounds
             | scanidrange COMMA indexcombexpr
             | scanidbounds COMMA indexcombexpr

scanidbounds: LT INT
            | GT INT
            | LE INT
            | GE INT
            | GE INT AMPERSAND LE INT
            | GT INT AMPERSAND LT INT

scanidrange: INT
           | INT TILDA INT
```

10.5 Time Selection Expression

```
token NUMBER  of type Integer
token FNUMBER of type Double

token TILDA
token LT
token GT
token COLON
token COMMA
token SLASH
token DOT
token STAR

type TEN timestatement
```

```
type TEN timeexpr
type TEN singletimeexpr
type TEN rangetimeexpr
type TEN upboundtimeexpr
type TEN lowboundtimeexpr
type Struct TimeFields yeartimeexpr
type Double FLOAT
type Vector<int> WNUMBER

timestatement: timeexpr

timeexpr: singletimeexpr
        | rangetimeexpr
        | lowboundtimeexpr
        | upboundtimeexpr
        | timeexpr COMMA timeexpr

WNUMBER: STAR
       | NUMBER

FLOAT: WNUMBER
     | FNUMBER {$$ = $1;}

singletimeexpr: yeartimeexpr

rangetimeexpr: yeartimeexpr TILDA yeartimeexpr
             | yeartimeexpr PLUS yeartimeexpr

lowboundtimeexpr: GT yeartimeexpr

upboundtimeexpr: LT yeartimeexpr

yeartimeexpr: WNUMBER SLASH WNUMBER SLASH WNUMBER SLASH WNUMBER
                COLON WNUMBER COLON FLOAT
            | WNUMBER SLASH WNUMBER SLASH WNUMBER SLASH WNUMBER
                COLON WNUMBER
            | WNUMBER SLASH WNUMBER SLASH WNUMBER SLASH WNUMBER
            | WNUMBER SLASH WNUMBER SLASH WNUMBER
```

```
            | WNUMBER SLASH WNUMBER SLASH WNUMBER
                COLON WNUMBER COLON FLOAT
            | WNUMBER COLON WNUMBER COLON FLOAT
            | WNUMBER COLON FLOAT
            | FLOAT
            | WNUMBER SLASH WNUMBER COLON WNUMBER COLON FLOAT
            | WNUMBER SLASH WNUMBER COLON WNUMBER
            | WNUMBER SLASH WNUMBER
```

10.6 UV-distance Selection Expression

```
token UNIT     of type String
token FNUMBER of type Double
token COLON
token COMMA
token PERCENT

type Double fnumwithunits
type TEN uvwdiststatement
type TEN uvwdistexprlist
type TEN uvwdistexpr

uvwdiststatement:uvwdistexprlist

uvwdistexprlist: uvwdistexpr
               | uvwdistexprlist COMMA uvwdistexpr

fnumwithunits: FNUMBER
             | FNUMBER UNIT

uvwdistexpr: fnumwithunits
           | FNUMBER TILDA fnumwithunits
           | LT fnumwithunits
           | GT fnumwithunits
           | fnumwithunits COLON FNUMBER PERCENT
```