ESCS schedule structure

	Name	Date	Signature
Authors	Righini S (IRA) Orlati A (IRA) Bartolini M (IRA)	02/02/15	

DOCUMENT CHANGE RECORD				
Issue No.	**Issue date**	**No. of pages**	**Pages changed, added, deleted**	**Description of change**
01	02/02/15	13	-	Issue 01
02	12/08/15	12	-	****Issue aligned to ESCS 0.5**** Addition of spectroscopy-related features. Overall rearrangement of contents. MBFITS-related info temporarily removed as this output format is not yet implemented.

1 Introduction

A schedule is a set of files where all the geometry/timing/frequency details of a sequence of data acquisitions are specified, according to a syntax that enables ESCS to read and execute them.
In order to generate the schedules, for the most common observing modes in continuum and spectroscopy, a tool called *ScheduleCreator* is available (see dedicated documentation).

For regular observations, **users should not edit the schedule files**: for most of the applications, they can totally ignore what is written inside them. Only expert users, wanting to customize their observations in unusual or complex ways, should access the schedules and edit them, or create schedules from scratch using their own tools.
For this purpose, the following sections describe the general scheme conceived for the observations and the content of the files composing the schedule.

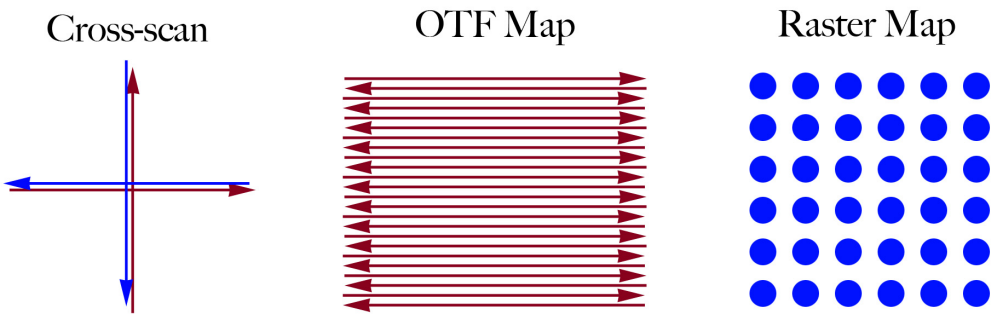
2 Scans vs subscans

We define the *scans* and *subscans* composing the observing session as follows:

Scan
It is the lowest level object normally used by an observer. *It is a sequence of one or more subscans that share a single goal*: for instance cross-scans and maps involve a pattern of subscans. Whether OTF maps mosaicing observations are considered a single scan or a series of scans is rather a matter of how the user would like to define it. In our implementation each map is considered a scan.

Subscan
it is the minimal amount of data acquisition that can be commanded at the script language level. It is highly desirable that it is a simple enough element. For example, it is the single OTF "line" of a map or of a cross-scan.

The figure below visually represents what cross-scans, OTF maps and raster maps are.



In the case of cross-scan, a subscan is a single arrow (a line across the target), four arrows – i.e. two full crosses – constitute the schema which might be repeated as many times as needed within the scan.
For OTF maps, the subscan is again the single arrow, and the scan coincides with the whole map obtained with lines along one axis only (e.g. along RA or Dec). For raster maps, which are based on discrete acquisitions, each point is a subscan, and the final map constitutes the scan.

When choosing FITS as the data output format, **a distinct FITS file is produced for each subscan** listed in the schedule. Details on the file production are given in the next section, where the .scd component of the schedule is described.

3 Schedule files

The present release of the system requires 4 files:

- **.scd** file: it holds the sequence of scans/subscans to be performed, including some setup parameters which can be either assigned to the whole schedule or to individual subscans
- **.lis** file: it lists the spatial configuration of the single subscans composing the observation; it also contains the target details in case of scpectral observations (target radial velocity, etc…)
- **.cfg** file: it contains the frontend configuration and other procedures to be used in the initialization phase (if any) and in the pre-scan/post-scan operations (if any). Any ESCS command can be inserted in these procedures. Users are warned, however, that their employment might not be necessarily useful: pay attention to the meaningfulness of their insertion within the schedules
- **.bck** file: it is devoted to the backend setup

3.1 SCD file

This is the main schedule file, sequentially listing the scans/subscans to be executed.
Values are all TAB-separated.
The header must contain the following keywords, including the final colon:

PROJECT:	user-defined label for the project. It will end up in the output filename.
OBSERVER:	name of the observer. It will end up in one FITS file header.
SCANLIST:	name of the LIS file to be used.
PROCEDURELIST:	name of the CFG file to be used.
BACKENDLIST:	name of the BCK file to be used.
MODE:	schedule mode: SEQ for sequential schedules, LST for time-based schedules. If SEQ, then a tab-separated start LST time can be specified as HH:MM:SS. If LST, then a tab-separated value can indicate how many times the schedule must be run
[SCANTAG:	number for the initial scan, optional, default is 1]
[INITPROC:	name of the initialization procedure (contained in the CFG file), optional]

Then, the scans/subscans are listed.
Each scan is introduced by a line starting with “SC:” followed by some scan-level information:

SC: Scan# ScanLabel BCKProcedure:WriterName [ScanLayoutName]

Scan#	is the number for the scan. It must be unique in the schedule. Scan numbers must be incremental but do not need to be sequential.
ScanLabel	will be included in the output filename.
BCKProcedure	is the name of a valid procedure listed in the BCK file.
WriterName	is the name of the output data writer (MANAGEMENT/FitsZilla or MANAGEMENT/MBFitsWriter).
ScanLayoutName	is the name of the layout selected from the DAT file. It should be omitted, as it is useless, when FITS files are chosen for data output.

After the scan setup, all the subscans composing that scan are specified, like:

Scan#_subscan# Duration SubscanID PreProcedure PostProcedure

Scan#_subscan#	is the sequential number for the subscan, e.g. 1_1, 1_2, etc…
Duration	is the subscan duration (seconds). For OTF subscans it must coincide with the duration declared in the LIS file
SubscanID	is the ID for the subscan to be executed, as reported in the LIS file
PreProcedure	is the name of the procedure, enclosed in the CFG file, to be performed in the POST-subscan phase. Parameters can be passed as name=value.
PostProcedure	is the name of the procedure, present in the CFG file, to be performed in he post-subscan phase. Parameters can be passed as name=value.

In case of a **sequential schedule**, scans/subscans are not associated to specific execution times, so they are carried out sequentially following an “as soon as possible” approach.
Here follows an example of sequential schedule where the chosen output file format is **FITS**.

```
PROJECT:        Test3c295
OBSERVER:       John Doe
SCANLIST:       Test3c295.lis
PROCEDURELIST:      Test3c295.cfg
BACKENDLIST:        Test3c295.bck
MODE: SEQ
INITPROC:       INIT

SC:     1       3c295   300_40:MANAGEMENT/FitsZilla
1_1     0.0     1       NULL    POSTTSYS
1_2     14.0    5       NULL    POST
1_3     14.0    6       NULL    POST
1_4     14.0    7       NULL    POST
1_5     14.0    8       NULL    PROC_WAIT=1

SC:     2       3c295   730_20:MANAGEMENT/FitsZilla
2_1     0.0     1       NULL    POSTTSYS
2_2     14.0    5       NULL    POST
2_3     14.0    6       NULL    POST
2_4     14.0    7       NULL    POST
2_5     14.0    8       NULL    PROC_WAIT=1
```

Please notice that **sequential schedules run ad libitum**, as long as the targets are above the horizon and the user does not input a *stopSchedule* command.

It is possible to write **sidereal-time-based schedules**, assigning the ‘LST’ value to the header keyword ‘MODE’, followed by the number of repetitions foreseen for the schedule (1 means that, when the schedule has completed one run, it stops). It is then necessary to add a column to the SCD file, where the LST start times for the individual subscans are provided. This feature is not included in the present release of the *schedulecreator*, so **this schedule version can be obtained only editing a sequential schedule**, or using custom tools.

The single subscan line then becomes:

Scan#_subscan# StartLST Duration SubscanID PreProcedure PostProcedure

Here is an example of time-based schedule:

```
PROJECT:        Test3c295
OBSERVER:       John Doe
SCANLIST:       Test3c295.lis
PROCEDURELIST:      Test3c295.cfg
BACKENDLIST:        Test3c295.bck
MODE: LST       1
INITPROC:       INIT

SC:     1       3c295   300_40:MANAGEMENT/FitsZilla
1_1     12:23:35.0      0.0     1       NULL    POSTTSYS
1_2     12:23:40.0      14.0    5       NULL    POST
1_3     12:24:00.0      14.0    6       NULL    POST
1_4     12:24:20.0      14.0    7       NULL    POST
1_5     12:24:40.0      14.0    8       NULL    POST

SC:     2       3c295   730_20:MANAGEMENT/FitsZilla
2_1     12:26:55.0      0.0     1       NULL    POSTTSYS
2_2     12:27:00.0      14.0    5       NULL    POST
2_3     12:27:20.0      14.0    6       NULL    POST
2_4     12:27:40.0      14.0    7       NULL    POST
2_5     12:28:00.0      14.0    8       NULL    POST
```

3.2 LIS file

This file lists all the spatial subscan configurations employed within the schedule, one for each line. They do not need to follow the execution sequence in which they are called by the SCD schedule: the first column gives a unique incremental ID (not necessarily sequential: gaps are allowed) to be included in the calls inside the SCD file. **Fields are TAB-separated**.

```
# 3C295
1   SIDEREAL    TSys        EQ      212.8360d 52.2025d  2000.0      -EQOFFS 0.0d    -0.35d
2   SIDEREAL    TSys        EQ      212.8360d 52.2025d  2000.0      -EQOFFS 0.0d    0.35d
3   SIDEREAL    MySource    GAL     200.3232d 45.1221d  -GALOFFS            0.0d    0.0d
4   SIDEREAL    OffSource   GAL     200.3232d 45.1221d  -GALOFFS            -1.0d   0.0d
5   OTF         3c295       212.8360d 52.2025d  0.0d    0.7d    EQ  EQ  LON CEN INC 14.0    -EQOFFS 0.0d    0.0
6   OTF         3c295       212.8360d 52.2025d  0.0d    0.7d    EQ  EQ  LON CEN DEC 14.0    -EQOFFS 0.0d    0.0
7   OTF         3c295       212.8360d 52.2025d  0.7d    0.0d    EQ  EQ  LAT CEN INC 14.0    -EQOFFS 0.0d    0.0
8   OTF         3c295       212.8360d 52.2025d  0.7d    0.0d    EQ  EQ  LAT CEN DEC 14.0    -EQOFFS 0.0d    0.0
```

Different subscan types can be used: SIDEREAL, OTF, OTFC and SKYDIP.

3.2.1 SIDEREAL subscans

They are used for tracking and on-off acquisitions: the antenna points to the specified position.
The LIS line is composed by:

ID = unique ID for the subscan configuration
TYPE = subscan type label, in this case 'SIDEREAL'
TARGET = label for subscan target/content
FRAME = frame for the coordinates to follow. Options: 'EQ', 'HOR' or 'GAL'
LONGITUDE = target longitude, following the generally allowed longitude formats
LATITUDE = target latitude, following the generally allowed latitude formats
EPOCH = for EQ coordinates only. '-1' means the coordinates are precessed to date.
OFFSET LABEL = [opt] frame for the offsets to follow. Options: '-EQOFFS', '-HOROFFS' or '-GALOFFS'
LON OFFSET = [opt] longitude offset (degrees, with 'd' suffix)
LAT OFFSET = [opt] latitude offset (degrees, with 'd' suffix)

→**Notice**: the offsets frame can be freely chosen, regardless of the frame describing the target coordinates. These offsets **sum up to the overall offsets** that might have been defined by the users with the *radecOffsets*, *azelOffsets* and *lonlatOffsets* commands. By default, **subscan-level offsets are zeroed any time a new subscan is commanded**, and new offsets (if any is specified in the LIS line) take over.

Though the definition SIDEREAL clearly implies the tracking of a celestial source, a "degenerate" use of this subscan type is given by *beam-parking* observations: when users want to acquire data in a fixed Az,El position, they can use SIDEREAL subscans where the coordinate frame is 'HOR'. Please notice that, though this observing mode implies no antenna motion, it fully corresponds to the execution of a schedule with scans/subscans as concerns data acquisition, so the mount must be in *tracking* mode (and not in *preset* mode) in order to perform this kind of observation.

3.2.2 OTF subscans

On-the-fly subscans are paths on the sky run at constant speed while acquiring data.

The LIS line is composed by:

ID = subscan unique ID
TYPE = subscan type, in this case 'OTF'
TARGET = label for target
LON1 = for DESCR='SS' (later keyword), longitude (*) of the scan starting point.
For DESCR='CEN', longitude (*) of the subscan central point.
LAT1 = for DESCR='SS' (later keyword), latitude (!) of the scan starting point. for DESCR='CEN', latitude (!) of the subscan central point.
LON2 = for DESCR='SS' (later keyword), longitude (*) of the subscan ending point. for DESCR='CEN', whole longitude span (*) of the subscan.
LAT2 = for DESCR='SS' (later keyword), latitude (!) of the subscan ending point. for DESCR='CEN', whole latitude span (!) of the subscan.
FRAME = coordinate frame relative to the previously specified lon-lat coordinates:
'EQ' = Equatorial J2000.0 (longitude = RA, latitude = Dec)
'HOR' = Horizontal (longitude = azimuth, latitude = elevation)
'GAL' = Galactic (longitude = l, latitude = b)
sFRAME = coordinate frame along which the scan is performed. It must be equal to FRAME, apart from a single case: if FRAME is 'EQ' and the scan description is 'CEN' (see next keywords), then sFRAME can also be 'HOR', which means that Az-El scans will be performed across a sidereal position. This is usually exploited for pointing calibration campaigns.
GEOM = scan geometry:
LON = constant longitude
LAT = constant latitude
GC = great circle arc (only for DESCR='SS')
DESCR = scan description:
SS = start and stop positions
CEN = center position + scan span
DIR = scan direction (ignored if GEOM='CG'):
INC = the varying coordinate increases
DEC = the varying coordinate decreases
DURATION = scan actual duration in seconds (acceleration/deceleration ramps excluded)
offFRAME = [opt] frame for the user-defined offsets to be added to the lon-lat coordinates specified (the leading dash ' –' is compulsory):
-EQOFFS = Equatorial
-HOROFFS = Horizontal
-GALOFFS = Galactic
At present offFRAME must be equal to sFRAME, which means that it is not possible to perform scans in an exotic way like scanning along the galactic longitude while applying equatorial offsets, etc... but it is possible to apply Az,El offsets to Az,El scans across a sidereal (equatorial) position.

LONOFF = [opt] longitude offset, in degrees, which can be specified as dd.dd'd' (decimal format, can be positive or negative, notice the 'd' suffix) or dd:mm:ss.s. It is meant to be "on sky", i.e. the actual span, in practice ∆lon x cos(lat).
LATOFF = [opt] longitude offset, in decimal degrees (notice the 'd' suffix).

→**Notice**: these offsets **sum up to the overall offsets** that might have been defined by the users with the *radecOffsets*, *azelOffsets* and *lonlatOffsets* commands. By default, **subscan-level offsets are zeroed any time a new subscan is commanded**, and new offsets (if any is specified in the LIS line) take over.

Examples of valid OTF subscans:

```
1  OTF  Source1  310.256d  30.231d  310.256d  30.931d  EQ   EQ   LON  SS   INC  14.0
2  OTF  Source1  310.256d  30.231d  0.0d      0.7d     EQ   EQ   LON  CEN  INC  14.0
3  OTF  Source2  12:45:12h 18:12:21.1 0.7d    0.0d     EQ   HOR  LAT  CEN  INC  14.0  -HOROFFS  -1.0d  0.0d
4  OTF  Source3  21.738d   88.205d  0.7d      0.0d     GAL  GAL  LAT  CEN  DEC  14.0  -GALOFFS  0.0d   1.0d
```

Subscans #1 and #2 are totally equivalent, as they only differ in the decription: the first gives the start-stop boundaries of the subscan, while the second expresses the same in terms of center+span.
Subscan #3 is a horizontal subscan (in particular it is 'LAT'=constant elevation), executed across a sidereal position, in this case expressed with sexagesimal coordinates, with -1 degree of azimuth offset.
Subscan #4 is performed in the Galactic frame, but with an offset of +1 degree in latitude. Notice that the subscan direction is 'DEC', so the subscan will be performed 'backwards', i.e. with longitude decreasing along the execution.

3.2.3 OTFC subscans

These OTF acquisitions are performed using an externally-defined target position.
The target is recovered from a separate subscan, whose ID is specified among the OTFC parameters:

ID = subscan unique ID
TYPE = subscan type, in this case 'OTFC'
TARGET_ID = ID of the subscan where the target position is defined
SPAN = span of the subscan (degrees)
FRAME = target coordinate frame ('EQ' or 'GAL')
sFRAME = coordinate frame along which the scan is performed ('EQ', 'HOR' or 'GAL')

GEOM = scan geometry:
LON = constant longitude
LAT = constant latitude
DIR = scan direction:
INC = the varying coordinate increases
DEC = the varying coordinate decreases
DURATION = scan actual duration in seconds (acceleration/deceleration ramps excluded)

Example of usage of OTFC subscans:

```
1  SIDEREAL          3c147
2  SIDEREAL          MySource          EQ     12:00:00h          30:00:00  2000.0
3  OTFC   1   1.0d   EQ   EQ   LAT  INC  14.0
4  OTFC   1   1.0d   EQ   EQ   LON  INC  14.0
5  OTFC   2   2.0d   GAL  GAL  LAT  INC  28.0
6  OTFC   2   2.0d   GAL  GAL  LON  INC  28.0
```

Subscans #1 and #2 are simple SIDEREAL subscans; notice that, as the first one invokes a target which is listed in the system catalogue (Appendix E), the target coordinates are not specified.
Numbers #3 and #4 are OTFC subscans centered on the position inserted in subscan #1; they grab the EQ coordinates of 3c147 from the catalogue and respectively perform a Dec and a RA subscan across the source, each spanning 1 degree in 14 seconds.
Subscans #5 and #6, instead, refer to the second SIDEREAL, the one devoted to 'MySource'. The target coordinates in subscan #2 are given in the Equatorial frame, however the OTFC calls them in the Galactic frame: in this case the target Equatorial coordinates are converted to Galactic coordinates and then passed to the OTF component of the system.

→ Notice: when a SIDEREAL subscan is used as a reference for an OTFC subscan, the offsets specified in the SIDEREAL one, if any, are ignored.

The present usage of OTFC subscans looks convoluted: it is mainly conceived for observations of moving targets (with the employment of an ephemeris generator component, under development).

3.2.4 SKYDIP subscans

A skydip can be achieved by properly setting a normal OTF subscan in the HOR frame, but this requires the user to already fix the azimuth position at which the skydip must be performed.
A more dynamical solution, described in the following paragraph, allows the observer to schedule a skydip in the nearbies of a given source.
To achieve this, here is an example of the lines which must be inserted in the .LIS file:

```
1 SIDEREAL MySource EQ  12:00:00h  30:00:00  2000.0  -HOROFFS  -1.0d  0.0d
2 SKYDIP    1    20.0d   90.0d   300.0   -HOROFFS  -1.0d  0.0d
```

The first line is a normal SIDEREAL subscan, pointing to an offset position w.r.t. a certain source of given celestial coordinates.
The second line is composed by:

ID = subscan unique ID
TYPE = subscan type, in this case 'SKYDIP'
REFERENCE_SIDEREAL = subscan ID identifying the reference SIDEREAL position
START_EL = elevation of skydip starting point (degrees, 0-90), with "d" suffix
STOP_EL = elevation of skydip ending point (degrees, 0-90), with "d" suffix
DURATION = subscan duration (seconds)
offFRAME = use -HOROFFS only
LONOFF = longitude offset (degrees), with "d" suffix
LATOFF = latitude offset (degrees), with "d" suffix (usually 0.0d)

The corresponding subscan definition in .SCD file would be:

```
SC: 1 MySource Skydip STD:MANAGEMENT/FitsZilla
1_1     0.0     1       NULL   POSTTSYS
1_2     300.0   2       NULL   POST
```

Where PROCEDURE_TSYS and POST are proper procedures written in the .CFG file (see next paragraph).
The result of the combination of the two actions is: the telescope goes off of 1.0° in azimuth with respect the target MySource, a tsys is measured. Then the current azimuth of the source (minus 1 degree) is used as the reference azimuth to perform the skydip, spanning beween 20° and 90° of elevation in 300 seconds.

3.3 CFG file

This file lists the (optional) configuration parameters for the frontend frequency management and for the execution of some procedures that the users might want to run before or after a scan.
The first field is the name of the procedure which is user-defined and must be unique in the file. Names are case sensitive. It is suggested to use all-caps names, so that any name clash with ESCS commands is impossible.
The open bracket must lie on the same line of the procedure name. Between brackets the configuration commands must be provided one for each line. The .CFG file can contain as many procedures as needed.
An example of a possible content of the .CFG file:

```
INIT{
    setLO=5600
    device=0
}

LOW_FREQ{
    setLO=5650
    device=0
}

HI_FREQ{
    setLO=5700
    device=0
}

POST{
    getTpi
}

POSTTSYS{
    wait=1.000
    tsys
}

PROC_WAIT(1){
    wait=$0
}
```

Observers might create a procedure like *INIT* (again, the name is user-defined), conceived to be called only once, when the schedule is loaded. One useful command to be inserted inside this call is the “setLO” one, to specify the local oscillator frequency value (MHz): this frequency setup can be manually performed during the overall system setup phase, but it also can be changed for each schedule inserting this command in the initialization procedure, or even at subscan level as shown in procedures *LOW_FREQ* and *HI_FREQ* to be used in the pre-scan phase. **For spectroscopy, do NOT use the “setLO” command, as the local oscillator frequency is computed by the system in order to properly observe the wanted line, whose actual sky frequency depends on date and time.**

The next procedures shown above are meant to be called before or after the execution of the actual subscan, and can be named as the user prefers: remember that their name must be correspondingly called inside the .SCD file (see dedicated section).
In the above example, if the procedure *POSTTSYS* is called after a certain subscan, a T_{sys} will be measured 1 second after the antenna has closed the subscan acquisition and before the next subscan is commanded. Please notice that these commands are NOT time-based: they will be executed sequentially.

Users can define procedures accepting one or more arguments, to be passed when the procedures are called. One example is the *WAIT* procedure: after its name, the number in () brackets specifies how many arguments it has. Inside this procedure, there is only a *wait* command, where the “$0” is a reference for the value that will be passed at runtime.

Ideally, any ESCS command can be inserted into these procedures. This does NOT mean it should be done, as many of them have no useful role within a schedule – on the contrary, their effects might be detrimental to the results. Users are warned that the execution of “creative” schedules might lead to unexpected or unwanted results.

Commands can be temporized, i.e. a specific UT time can be associated to them, in order to command their execution in a given moment. This is accomplished appending a time indication in the form "@DOY-HH:MM:SS", for example:

tsys@124-13:44:23

Again, this opportunity must be exploited *cum grano salis*.

3.4 BCK file

The content of this file is devoted to the backend configuration. As in the .CFG file, it lists the procedures to be called within the .SCD schedule.
As the following example shows, each procedure must have a unique name and make reference to the selected backend (in this case: BACKENDS/TotalPower). The open bracket must line in the same line of the procedure name, then the backend-related commands must be inserted – one per line.

```
STD:BACKENDS/TotalPower {
    setSection=0,*,2000.0,*,*,0.000025,*
    setSection=1,*,2000.0,*,*,0.000025,*
    integration=40
    enable=1;1
}

300_40:BACKENDS/TotalPower {
    setSection=0,*,300.0,*,*,0.000025,*
    setSection=1,*,300.0,*,*,0.000025,*
    integration=40
    enable=1;1
}

730_20:BACKENDS/TotalPower {
    setSection=0,*,730.0,*,*,0.00005,*
    setSection=1,*,730.0,*,*,0.00005,*
    integration=20
    enable=1;1
}
```

Notice the 'enable' command, which positionally specifies the sections that are meant to be acquiring data (0 equals to 'off', 1 to 'on'). If, for example, only the feeds 0 and 5 of the MF receiver are required to observe, the command for the TPB would be: enable=1,1,0,0,0,0,0,0,1,1,0,0
It is important to give the *integration* command *after* the *setSection* one, when the integration value differs from the sampling time commanded by *setSection*.

3.5 Spectral line observations

When spectroscopy is concerned, the relevant **source parameters** can be specified, **for any subscan type**, by adding the following optional elements to the needed .**LIS file** line:

RVEL LABEL = [opt] in case the source radial velocity needs to be specified, the '-RVEL' key must be written, followed by a three-value argument
RADIAL VELOCITY = [opt] km/s, unless the VELOCITY DEFINITION keyword is set to 'Z' (in that case it is a dimensionless quantity)
VELOCITY FRAME = [opt] frame for the radial velocity. Options: BARY, LSRK, LSRD, LGRP, GALCEN, TOPOCEN
VELOCITY DEFINITION = [opt] radial velocity definition. Options: RD (Radio), OP (Optical), Z (Redhsift)

Example of LIS line:

```
1   SIDEREAL MySource  GAL   200.3232d 45.1221d  -GALOFFS   0.0d   0.0d   -RVEL   112.223   LSRK   RAD
```

The observed **line rest frequency**, instead, can be specified in the INIT or PRESCAN procedures inside the .**CFG file**, using the following command:

```
restFrequency=[freq1];…;[freqN]
```

where values must be expressed in MHz. Providing only one frequency value means that all the N sections available for the receiver in use will be set to the same frequency, otherwise it is compulsory to specify all the N values.

In order to let the system compute the **observed frequency to be centered** in the band(s) at runtime, thus taking into account the proper Doppler effects, a specific **command** must be inserted in the PRESCAN procedures of the .**CFG file**, after having provided the line rest frequency in the same procedure or in the general INIT one:

```
fTrack=[dev]
```

where [dev] is the device in charge of performing the frequency tuning:

- LO: only the front-end local oscillator is moved
- ALL: first of all the front-end local oscillator is tuned, then the back-end - if it allows such a sub-tuning - also performs a further frequency adjustment, in order to centre the line(s) in the various sections. This option is useful in case multiple rest frequencies are to be observed, yet a complete success cannot be guaranteed as the dopplered frequencies might turn out to be incompatible with the section bandwidths and the LO ranges. In case at least one line lies outside the RF band of the receiver or the back-end input bandwidth, an error rises.

Example of .CFG procedures:

```
INIT_SPECTRAL{
    restFrequency=5678.9992
}

PRESCAN_SPECTRAL{
    fTrack=ALL
    device=0
}
```