AIPS++ User Reference Manual - NRAO Package

AIPS++ Group, eds

August 1, 2006

The AIPS++ consortium may be reached by email at aips2-request@nrao.edu. The postal address is: AIPS++ Consortium, c/o NRAO, 520 Edgemont Rd., Charlottesville, Va. 22903-2475 USA.

Contents

Chapter 1

NRAO

NRAO-related modules and tools

include "nrao.g"

Description
Package for NRAO-related functionality

1.1 gbt - Module

Module for GBT data processing
include "gbt.g"

Description
Various functions and tools for processing GBT logs and data. Most can be invoked either from within Glish or from the UNIX command line. Both forms are documented here.
Functions

gbtlogfiller	Join GBT logs into a single AIPS++Table
gbtlogview	View GBT logs

Tools

dcr	A tool for analyzing DCR data from the GBT.
gbtmsfiller	A tool for filling GBT data to an aips++ MeasurementSet.

1.1.1 dcr - Tool
Package nrao

Module gbt

A tool for analyzing DCR data from the GBT.
include "dcr.g"

Constructors

dcr	Construct dcr tool

Functions

baselinefit	Fit a baseline
done	Close and exit the DCR tool
focus	Fit a parabola to a focus observation
focusScan	Reduce a single focus scan.
gauss	Fit a gaussian function to the data.
getGO	Return the GO keywords
get_tant	Returns Antenna Temperature for a given scan
getscan	Return data for the given scan
guessmode	Guess the switching signal mode
listscans	List scan numbers
plot_RA_Dec	Plot Dec vs. RA
plot_dap_time	Plot a data associated parameter (DAP) vs. time
plot_focus_time	Plot focus against time
plot_gain_time	Plot relative gain (normalized counts/K) vs. time - not yet implemented
plot_phase_dec	Plot data counts vs. Dec for a given phase
plot_phase_ra	Plot data counts vs. RA for a given phase
plot_phase_time	Plot data counts vs. time for a given phase
plot_sidelobe	Plot sidelobes
plot_tant_Dec	Plot Antenna Temperature vs. Dec
plot_tant_RA	Plot Antenna Temperature vs. RA
plot_tant_time	Plot Antenna Temperature vs. time
plot_tsrc_time	Plot Tsrc against time
plotscans	Plot several scans contiguously
point1	Reduce a 1 scan pointing observation
point2	Reduce a 2 scan pointing observation
point4	Reduce a 4 scan pointing observation
scanSummary	Print a summary of the scans in the GUI message window
test_srp	Test whether a specified scan, receiver, phase exists
tip	Reduce a tipping scan.
tsys	Calculate the system temperature
type	Return the type of this tool.

Description
This `tool` is an in-house suite of functions intended to be used exclusively with digital continuum receiver (DCR) data from the Green Bank Telescope (GBT). The `tool` consists of 2 GUI components: A panel of buttons which offers an interface to some of the common functionality, and a plotter window which displays results when applicable. The complete functionality of the dcr tool is available from the command line interface. The tool is aimed at satisfying the requirements for commissioning the GBT, primarily, but it may find use as a general purpose continuum analysis tool as well.

dcr - Constructor

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Construct dcr tool

Synopsis
dcr(filename)

Description
This constructs the `tool` which provides numerous functions for analyzing DCR data. The constructor requires the name of a data file which must be an AIPS++ Measurement Set. To access data from a different data file it is necessary to open a new instance of the tool.

Arguments

filename	filename of an AIPS++ Measurement Set which contains the data Allowed:string Default: unset

Returns
Tool

---

dcr.getscan - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Return data for the given scan

Synopsis
getscan(scan, getFocus)

Description
The most fundamental routine in the dcr tool. This returns a record containing information about the selected scan, including header info and the data. It also sets up variables which are used internal to the dcr tool by other routines. The focus values are not included by default in order to improve efficiency.

Arguments

scan	Scan number Allowed:Int
getFocus	flag to determine whether focus values will be retrieved Allowed:Boolean Default: F

Returns
record

---

dcr.listscans - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

List scan numbers

Synopsis
listscans()

Description
Returns an array containing the scan numbers.

Returns
integer array

dcr.done - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Close and exit the DCR tool

Synopsis
done()

Description
Closes the DCR tool and removes it from memory.

Returns
boolean

---

dcr.getGO - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Return the GO keywords

Synopsis
getGO(scan)

Description
The astronomer's interface to controlling the GBT (the GO program) produces FITS files containing keywords relevant to the observing setup. This function returns a record containing those keywords.

Arguments

scan	Scan number Allowed:Int

Returns
record

---

dcr.guessmode - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Guess the switching signal mode

Synopsis
guessmode(scan)

Description
By looking at the number and arrangement of phases, this routine makes a guess at the switching signal mode. It has been made redundant by an entry in the GO header which conveys the same parameter directly.

Arguments

scan	Scan number Allowed:Int

Returns
string

---

dcr.baselinefit - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Fit a baseline

Synopsis
baselinefit(xarray, yarray, ord, range, plotflag)

Description
This function provides a baseline fit to the given data. The residuals are returned.

Arguments

xarray	x values Allowed:double
yarray	y values Allowed:double
ord	Order of the polynomial Allowed:int
range	Channel range to be fit Allowed:array
plotflag	Plot the fit? Allowed:boolean Default: T

Returns
vector of residuals

---

dcr.focus - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Fit a parabola to a focus observation

Synopsis
focus(filename)

Arguments

filename	name of ascii file on disk Allowed:string

Returns
boolean

Description
Read an ascii file containing focus values from a sequence of point/focus combinations, and fit a parbola to find the peak value on the curve.

Example

```
include 'dcr.g'
md := dcr('GBTdata.ms')  # suppose we have this data set on disk
md.focus('focus.data')
```

where in this case the file focus.data might have this form:

```
1.2   15.5
1.3   15.7
1.4   16.0
1.5   16.1
1.6   15.9
1.7   15.6
1.8   15.4
```

---

dcr.get_tant - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Returns Antenna Temperature for a given scan

Synopsis
get_tant(scan, receiver, cal_value)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int Default: 1
cal_value	The calibration value for deriving counts/K Allowed:float Default: 1

Returns
boolean

Description
Returns an array of antenna temperatures.

---

dcr.gauss - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Fit a gaussian function to the data.

Synopsis
gauss(xarray, yarray, height, width, center, plotflag)

Arguments

xarray	X values Allowed:double
yarray	Y values Allowed:double
height	initial guess for height Allowed:float
width	initial guess for width Allowed:float
center	initial guess for center Allowed:float
plotflag	plot the result? Allowed:boolean Default: T

Returns
vector of residuals.

Description
This function fits a gaussian to the data. The residuals are returned.

---

dcr.plot_dap_time - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot a data associated parameter (DAP) vs. time

Synopsis
plot_dap_time(scan, colName)

Arguments

scan	Scan number Allowed:Int
colName	colName from the DAP subtable Allowed:string

Returns
boolean

---

dcr.plot_gain_time - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot relative gain (normalized counts/K) vs. time - not yet implemented

Synopsis
plot_gain_time(scan, receiver)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int

Returns
boolean

---

dcr.plot_RA_Dec - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot Dec vs. RA

Synopsis
plot_RA_Dec(scan)

Arguments

scan	Scan number Allowed:Int

Returns
boolean

---

dcr.plot_phase_ra - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot data counts vs. RA for a given phase

Synopsis
plot_phase_ra(scan, receiver, phase)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int
phase	Phase number Allowed:Int

Returns
boolean

---

dcr.plot_phase_dec - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot data counts vs. Dec for a given phase

Synopsis
plot_phase_dec(scan, receiver, phase)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int
phase	Phase number Allowed:Int

Returns
boolean

---

dcr.plot_phase_time - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot data counts vs. time for a given phase

Synopsis
plot_phase_time(scan, receiver, phase)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int
phase	Phase number Allowed:Int

Returns
boolean

---

dcr.plot_tsrc_time - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot Tsrc against time

Synopsis
plot_tsrc_time(scan, receiver, cal_value)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int
cal_value	The calibration value for deriving counts/K Allowed:float Default: 1

Returns
boolean

Description
Plot source temperature against time. This only works when the switching phases are set to switched power with cal.

---

dcr.plot_focus_time - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot focus against time

Synopsis
plot_focus_time(scan, param)

Arguments

scan	Scan number Allowed:Int
param	Focus parameter Allowed:string Default: SR_XP

Returns
boolean

Description
Plots a focus value against time. The focus parameter can be any of SR_XP, SR_YP, SR_ZP, SR_XT, SR_YT, or SR_ZT for gregorian focus observing, or ANTPOSPF, PF_FOCUS, or PF_ROTATION for prime focus observing.

---

dcr.focusScan - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Reduce a single focus scan.

Synopsis
focusScan(scan, receiver, cal_value, param, order, archive)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int Default: 0
cal_value	Cal value Allowed:float Default: 1
param	Focus parameter Allowed:string Default: SR_XP
order	Order of polynomial to fit Allowed:int Default: 2
archive	Archive the result? Allowed:boolean Default: F

Returns
boolean

Description
Plots a focus value against time. The focus parameter can be any of SR_XP, SR_YP, SR_ZP, SR_XT, SR_YT, or SR_ZT for gregorian focus observing, or PF_X, PF_FOCUS, or PF_ROTATION for prime focus observing. A polynomial is fit the to data, and the peak value from the fit is reported.

---

dcr.plot_tant_time - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot Antenna Temperature vs. time

Synopsis

plot_tant_time(scan, receiver, cal_value)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int Default: 1
cal_value	The calibration value for deriving counts/K Allowed:float Default: 1

Returns
boolean

---

dcr.plot_tant_RA - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot Antenna Temperature vs. RA

Synopsis
plot_tant_RA(scan, receiver, cal_value)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int Default: 1
cal_value	The calibration value for deriving counts/K Allowed:float Default: 1

Returns
boolean

---

dcr.plot_tant_Dec - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot Antenna Temperature vs. Dec

Synopsis
plot_tant_Dec(scan, receiver, cal_value)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int Default: 1
cal_value	The calibration value for deriving counts/K Allowed:float Default: 1

Returns
boolean

---

dcr.plot_sidelobe - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot sidelobes

Synopsis
plot_sidelobe(scan, receiver, basepct, bottom)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int Default: 1
basepct	Percent of scan to use for baseline Allowed:Int Default: 10
bottom	dB level to use for cutoff at bottom of plot Allowed:Int Default: -70

Description
Plots the sidelobes on the traditional dB scale. Best if used on a scan intended for this purpose.

Returns
boolean

---

dcr.plotscans - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Plot several scans contiguously

Synopsis
plotscans(bscan, escan, receiver, phase)

Arguments

bscan	beginning scan number Allowed:Int
escan	ending scan number Allowed:Int
receiver	Receiver number Allowed:Int
phase	Phase number Allowed:Int

Returns
boolean

---

dcr.point1 - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Reduce a 1 scan pointing observation

Synopsis
point1(scan, receiver, xaxis, cal_value, basepct, plotflag)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int
xaxis	1=RA, 2=dec Allowed:Int
cal_value	The calibration value for deriving counts/K Allowed:float Default: 1
basepct	percentage of scan to use on the edges for a baseline fit Allowed:Int Default: 10
plotflag	flag for plotting the data Allowed:Boolean Default: T

Description
Reduces a pointing observation made with the GBT. Baselines are fit from the two edges of the scan only. The parameter basepct specifies the percentage of the total scan used from each edge. For example, a basepct of 10 means that a tenth of the scan at each of the two edges is used in calculating the baseline. If xaxis=0, the function will attempt to determine whether RA or Dec is appropriate for the given scan.

Returns
boolean

dcr.point2 - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Reduce a 2 scan pointing observation

Synopsis
point2(scan, receiver, cal_value)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int
cal_value	The calibration value for deriving counts/K Allowed:float Default: 1

Description
Automated reduction of 2-scan pointing observations with the GBT.

Returns
boolean

dcr.point4 - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Reduce a 4 scan pointing observation

Synopsis
point4(scan, receiver, cal_value, plotflag)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int
cal_value	The calibration value for deriving counts/K Allowed:float Default: 1
plotflag	flag for plotting the data Allowed:Boolean Default: T

Description
Automated reduction of 4-scan pointing observations with the GBT.

Returns
boolean

---

dcr.test_srp - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Test whether a specified scan, receiver, phase exists

Synopsis
test_srp(scan, receiver, phase)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int
phase	Receiver number Allowed:Int

Description
This is used to test whether a given combination of scan, receiver, and phase

exist in the opened MeasurementSet. This returns T(rue) if the combination does NOT exist.

Returns
boolean

---

dcr.scanSummary - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Print a summary of the scans in the GUI message window

Synopsis
scanSummary()

Returns
boolean

---

dcr.tsys - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Calculate the system temperature

Synopsis
tsys(scan, receiver)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int

Returns
boolean

---

dcr.tip - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Reduce a tipping scan.

Synopsis
tip(scan, receiver)

Arguments

scan	Scan number Allowed:Int
receiver	Receiver number Allowed:Int

Returns
boolean

Description
Reduce a tipping scan. Uses least-squares to derive an opacity.

---

dcr.type - Function

Package	**nrao**
Module	**gbt**
Tool	**dcr**

Return the type of this tool.

Synopsis
type()

Returns
string

Description
This is required for the toolmanager to manage this tool. This function always returns "dcr".

1.1.2 gbtmsfiller - Tool

Package nrao
Module gbt

A tool for filling GBT data to an aips++ MeasurementSet.
include "gbtmsfiller.g"

Constructors

gbtmsfiller	Construct gbtmsfiller tool

Functions

backend	Get the current backend type
compresscalcols	Get the value of the compresscalcols flag.
dcbias	Get the specific value of the dcbias to use during the van Vleck correction.
done	Stop filling and close the tool.
fillall	Start filling all of the scans in the scan log
filllags	Get the value of the filllags flag.
fillnext	Start filling the next scan in the scan log
fillrawfocus	Get the value of the fillrawfocus flag.
fillrawpointing	Get the value of the fillrawpointing flag.
fixlags	Get the fixlags toggle value.
isattached	Does the filler have a valid project directory
maxscan	Get the current maximum scan number for data to be filled
maxtime	Get the current maximum time for data to be filled
minbiasfactor	Get the minbiasfactor value.
minscan	Get the current minimum scan number for data to be filled
mintime	Get the current minimum time for data to be filled
more	Are there more scans to process in the scan log?
msdirectory	Get the current directory of all output MSs
msrootname	Get the current msrootname
newms	Force a new MS for all currently filling backends.
object	Get the current object string to match
oneacsms	Get the value of the oneacsms flag.

project	Get the current project directory
setbackend	Limit the backends to fill to one type
setcompresscalcols	Set the flag which controls whether the calibration columns are compressed.
setdcbias	Set a specific bias to use during the van Vleck correction.
setfilllags	Set the flag which controlls whether the LAG_DATA data column is filled.
setfillrawfocus	Set the flag which controlls whether the raw focus values are saved (filled).
setfillrawpointing	Set the flag which controlls whether the raw pointings are saved (filled).
setfixlags	Toggle whether bad ACS lags should be fixed, if possible.
setmaxscan	Change the maximum scan number for data to be filled
setmaxtime	Change the maximum time for data to be filled
setminbiasfactor	Attempt to correct for the truncation error in the ACS samplers.
setminscan	Change the minimum scan number for data to be filled
setmintime	Change the minimum time for data to be filled
setmsdirectory	Change the directory of all output MSs
setmsrootname	Change the root name of all output MSs
setobject	Limit the fill to objects matching this string
setoneacsms	Set the flag which controlls whether a single ACS MeasurementSet is filled.
setproject	Set the project directory
setsmooth	Choose the type of smoothing for spectrometer data.
setusebias	Set the flag which controlls whether the bias is calculated during the van Vleck correction.
setusehighcal	Set the flag which controlls which CAL_TEMP is used from the receiver cal table.
setvv	Choose the type of vanVleck correction for spectrometer data.
smooth	Get the value of type of smoothing being used.
status	Report the status of the filler
type	Return the type of this tool.
update	Reopen and resync with the scan log
usebias	Get the value of the usebias flag.
usehighcal	Get the value of the usehighcal flag.
vv	Get the value of type of vanVleck correction being used.

Description

This `tool` is intended to be run by the GBT operators during normal operation of the telescope. Several MeasurementSets will typically be filled at the same time (one for each backend in use by each project during an observing session). The filling process is driven by the ScanLog.fits file for each project. This tool is under development and the exact operational details have not yet been finalized. It is anticipated that most interations with this `tool` will be done through its GUI.

Example

No example available yet.
This space reserved for comments about the example, when it becomes available.

gbtmsfiller - Constructor

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Construct gbtmsfiller tool

Synopsis
gbtmsfiller()

Returns
object

Description
This creates the gbtmsfiller `tool`. The resulting `tool` is not yet ready to fill. The project directory must first be specified.

Example

```
gbtfiller := gbtmsfiller()
gbtfiller.setproject('/home/gbtdata/AGBT02A_028_01')
# the filler is now ready to fill
gbtfiller.fillall()
```

The plan is that once a project is specified, subsequent filling will happen automatically when new data arrives. For the time being, the decision to fill to a new MS as opposed to continuing to append to the old MS will be left up to the operator. It is expected that reasonable heuristics will be developed to make this process automatic as well although the ability to force a new MS will continue to be available.

---

gbtmsfiller.isattached - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Does the filler have a valid project directory

Synopsis
isattached()

Returns
boolean

Description
Does the current project directory exist and is there a ScanLog.fits file there? When this function returns True, the filler is attached to a project and is ready to begin filling.

---

gbtmsfiller.fillall - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Start filling all of the scans in the scan log

Synopsis
fillall(async)

Arguments

async	Run this command asyncronously Allowed:boolean Default: F

Returns
boolean

Description
Start filling. All of the known data on disk at the time of the last update of the scan log is filled. When run asyncronously, this function returns before the filling has completed. Watch the log in that case to see the progress of the filling.

---

gbtmsfiller.fillnext - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Start filling the next scan in the scan log

Synopsis
fillnext(async)

Arguments

async	Run this command asyncronously Allowed:boolean Default: F

Returns
boolean

Description
Start filling. The next scan in the scan log is filled. It is anticipated that this function will be automatically called during normal operation whenever new data is written to disk (update is called first). When run asyncronously, this function returns before the filling has completed. Watch the log in that case to see the progress of the filling.

---

gbtmsfiller.update - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Reopen and resync with the scan log

Synopsis
update(async)

Arguments

async	Run this command asyncronously Allowed:boolean Default: F

Returns
boolean

Description
This should be called whenever the scan log FITS file has changed so that the filler can know about all of the data on disk. When run asyncronously, this function returns before the update has completed. Watch the log in that case to see the progress of the update.

---

gbtmsfiller.more - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Are there more scans to process in the scan log?

Synopsis
more()

Returns
boolean

Description
The answer to this question reflects the state of the scan log at the last update. If the scan log has since changed on disk, update must first be called in order that the filler can see those changes.

---

gbtmsfiller.status - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Report the status of the filler

Synopsis
status()

Returns
record

Description
Returns the status record.

status - Record	the status record
status.attached	is the filler attached Allowed: boolean
status.nextscan	pending scan number to be processed Allowed: integer
status.nextdmjd	pending DMJD value to be processed Allowed: double
status.nexttimestamp	pending scan timestamp to be processed Allowed: double
status.project	project being filled Allowed: string
status.msdirectory	The directory where any MSs are being written Allowed: string
status.object	object constraint Allowed: string
status.mintime	minimum time constraint Allowed: string
status.maxtime	maximum time constraint Allowed: string
status.minscan	minimum scan number constraint Allowed: integer
status.maxscan	maximum scan number constraint Allowed: integer
status.fillrawpointing	Are the raw pointings being saved? Allowed: boolean
status.fillrawfocus	Are the raw focus data being saved? Allowed: boolean
status.filllags	Are the raw lags being filled? Allowed: boolean
status.vv	The current vanVleck correction Allowed: string
status.smooth	The current smoothing function

	Allowed: string
status.usehighcal	Is the $HIGH_CAL$ being used? (old data only)
	Allowed: boolean
status.compresscalcols	Are the spectral line calibration columns compressed?
	Allowed: boolean
status.usebias	Is the bias being calculated for the ACS vanVleck correction?
	Allowed: boolean
status.backendtype	backend type constraint
	Allowed: string
status.oneacsms	Is a single ACS MS being filled?
	Allowed: boolean
status.dcbias	Current DC bias value.
	Allowed: double
status.minbiasfactor	Minimum bias factor.
	Allowed: integer
status.fixlags	Try and fix the ACS lags.
	Allowed: boolean
status.dcr	Status of the MS being filled with DCR data
	Allowed: backend status record
status.holo	Status of the MS being filled with Holography data
	Allowed: backend status record
status.sp	Status of the MS being filled with SpectralProcessor data
	Allowed: backend status record
status.acs	Record of status records for ACS (spectrometer) data
	Allowed: record

The ACS (spectrometer) status record has the following structure:

acs - Record	Record of status records for ACS data
acs.ABCD	Status of the MS being filled with all ACS data (banks A,B,C,D)
	Allowed: backend status record
acs.A	Status of the MS being filled with bank A ACS data
	Allowed: backend status record
acs.B	Status of the MS being filled with bank B ACS data
	Allowed: backend status record
acs.C	Status of the MS being filled with bank C ACS data
	Allowed: backend status record
acs.D	Status of the MS being filled with bank D ACS data
	Allowed: backend status record

Each backend has a status record (the ACS (spectrometer) backend has 5 status records, one for the single MS being filled with data from all banks and one for each possible bank, all grouped in the acs record). The backend status record looks like this:

backend status - Record	the backend status record
backend status.ms	The name of the MS being filled. This is unset if it is not being filled. Allowed: string
backend status.nrows	The total number of rows in that MS. This is -1 if it is not being filled. Allowed: integer

If the attached field is False, all of the other fields may not reflect the true state of the filler. The project, object, mintime, maxtime, minscan, maxscan, fillrawpointing, fillrawfocus, filllags, vv, smooth, usehighcal, compresscalcols, usebias, oneacsms and backendtype constraints can all be set via this DO interface. When the filler is at the end of the scan log (more() return False) then nextscan will have a value of -1 and nextdmjd and nextimestamp will have values of 0.0.

---

gbtmsfiller.setproject - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Set the project directory

Synopsis
setproject(project)

Arguments

project	The project directory Allowed:string

Returns
boolean

Description
Setting the project directory causes any currently open ScanLog.fits file as well as any MeasurementSets currently being filled to be closed. The next fill will open the new ScanLog.fits and begin filling information found there.

---

gbtmsfiller.project - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the current project directory

Synopsis
project()

Returns
string

Description
Get the current project directory known to the filler. This returns an empty string if no project has been set.

---

gbtmsfiller.setbackend - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Limit the backends to fill to one type

Synopsis
setbackend(backend)

Arguments

backend	The backend type to fill Allowed:string

Returns
boolean

Description

Possible types are ANY, HOLO, DCR, SP, and ACS. SPECTRALPROCESSOR is synonymous with SP and SPECTROMETER is synonymous with ACS. Any other types from that specified here will be ignored. The default, ANY, implies that all of the known types will be filled. This argument is case insensitive.

---

gbtmsfiller.backend - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the current backend type

Synopsis
backend()

Returns
string

Description
Get the current backend type being filled. The default is ANY, implying that any backends known to the filler are filled.

---

gbtmsfiller.setmsdirectory - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Change the directory of all output MSs

Synopsis
setmsdirectory(msdirectory)

Arguments

msdirectory	The directory of all output MSs Allowed:string

Returns
boolean

Description
The output MSs are all placed in this directory. This function allows you to change the default, which is the current directory.

---

gbtmsfiller.msdirectory - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the current directory of all output MSs

Synopsis
msdirectory()

Returns
string

---

gbtmsfiller.setmsrootname - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Change the root name of all output MSs

Synopsis
setmsrootname(msrootname)

Arguments

msrootname	The root name of all output MS Allowed:string

Returns
boolean

Description
The output MSs are named using this string plus the backend type plus a version number: msrootname_backendtype. This function allows you to change the default, which is the base name of the project directory. The backendtype is one of HOLO, DCR, SP, or ACS. If multiple ACS MeasurementSets are being filled. There will be an additional _A, _B, _C, or _D appended to the name as necessary.

---

gbtmsfiller.msrootname - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the current msrootname

Synopsis
msrootname()

Returns
string

---

gbtmsfiller.setmintime - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Change the minimum time for data to be filled

Synopsis
setmintime(mintime)

Arguments

mintime	The minimum time. Allowed:string

Returns
boolean

Description
Only data taken after mintime will be filled. The default is for there to be no minimum time and fill all data. If a sub-scan spans mintime (i.e. starts before mintime but ends after mintime) the entire sub-scan will be filled. The time string is specified using the FITS time convention:
YYYY-MM-DDTHH:MM:SS.sss.

---

gbtmsfiller.mintime - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the current minimum time for data to be filled

Synopsis
mintime()

Returns
string

---

gbtmsfiller.setmaxtime - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Change the maximum time for data to be filled

Synopsis

setmaxtime(mintime)

Arguments

mintime	The maximum time. Allowed:string

Returns
boolean

Description
Only data taken before maxtime will be filled. The default is for there to be no maximum time and fill all data. If a sub-scan spans maxtime (i.e. starts before maxtime but ends after maxtime) the entire sub-scan will be filled. The time string is specified using the FITS time convention:
YYYY-MM-DDTHH:MM:SS.sss.

---

gbtmsfiller.maxtime - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the current maximum time for data to be filled

Synopsis
maxtime()

Returns
string

---

gbtmsfiller.setminscan - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Change the minimum scan number for data to be filled

Synopsis
setminscan(minscan)

Arguments

minscan	The minimum scan number. Allowed:integer

Returns
boolean

Description
Only data having a scan number greater than or equal to minscan will be filled. Specifying a value of -1 (the default at startup) indicates that there is no minimum scan number.

---

gbtmsfiller.minscan - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the current minimum scan number for data to be filled

Synopsis
minscan()

Returns
integer

---

gbtmsfiller.setmaxscan - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Change the maximum scan number for data to be filled

Synopsis
setmaxscan(minscan)

Arguments

minscan	The maximum scan number. Allowed:integer

Returns
boolean

Description
Only data having a scan number less than or equal to maxscan will be filled. Specifying a value of -1 (the default at startup) indicates that there is no maximum scan number.

---

gbtmsfiller.maxscan - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the current maximum scan number for data to be filled

Synopsis
maxscan()

Returns
integer

---

gbtmsfiller.setobject - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Limit the fill to objects matching this string

Synopsis
setobject(object)

Arguments

object	The object string to match Allowed:string

Returns
boolean

Description
Only scans with objects matching this string will be filled. Simple wild cards involving * are allowed. If this string is just that wildcard (the default when this `tool` is created) then all objects will match.

---

gbtmsfiller.object - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the current object string to match

Synopsis
object()

Returns
string

---

gbtmsfiller.setfillrawpointing - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Set the flag which controlls whether the raw pointings are saved (filled).

Synopsis
setfillrawpointing(fillrawpointing)

Arguments

fillrawpointing	When true, raw pointings are filled. Allowed:boolean Default: F

Returns
boolean

Description
The GBT Antenna FITS file contains information on the telescope pointings. The pointing information is normally sampled 10 times a second. The filler computes a mean pointing direction over each integration and stores that value in the DIRECTION column of the POINTING subtable. When the fillrawpointing flag is F(alse) (the default value) the raw, unaveraged, pointings are not saved in the MeasurementSet in order to save space. This behavior can be changed by setting this flag to T. When this flag is true, the raw pointings can be found in the NRAO_GBT_POINTING subtable. This decision must be made before the MeasurementSet is filled.

---

gbtmsfiller.fillrawpointing - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the value of the fillrawpointing flag.

Synopsis
fillrawpointing()

Returns
boolean

---

gbtmsfiller.setfillrawfocus - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Set the flag which controlls whether the raw focus values are saved (filled).

Synopsis
setfillrawfocus(fillrawfocus)

Arguments

fillrawfocus	When true, raw focus values are filled. Allowed:boolean Default: F

Returns
boolean

Description
The GBT Antenna FITS file contains information on the telescope focus parameters. The focus information is normally sampled 10 times a second. The filler computes the mean focus parameters over each integration and stores that value in the NRAO_GBT_MEAN_FOCUS subtable (future versions of the filler will likely associate this information more closely with the FEED subtable). When the fillrawfocus flag is F(alse) (the default value) the raw, unaveraged, focus values are not saved in the MeasurementSet in order to save space. This behavior can be changed by setting this flag to T. When this flag is true, the raw focus values can be found in the NRAO_GBT_FOCUS subtable. This decision must be made before the MeasurementSet is filled.

---

gbtmsfiller.fillrawfocus - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the value of the fillrawfocus flag.

Synopsis

fillrawfocus()

Returns
boolean

---

gbtmsfiller.setfilllags - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Set the flag which controlls whether the LAG_DATA data column is filled.

Synopsis
setfilllags(filllags)

Arguments

filllags	When true, the LAG_DATA column is filled. Allowed:boolean Default: F

Returns
boolean

Description
This flag is only used to control filling of data from the ACS (spectrometer). Normally, when this flag is false, the raw lag data are only used to generate the FLOAT_DATA column. When this flag is true, the raw data are also filled to the LAG_DATA column. The FLOAT_DATA column will continue to be filled in that case.

---

gbtmsfiller.filllags - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the value of the filllags flag.

Synopsis
filllags()

Returns
boolean

---

gbtmsfiller.setvv - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Choose the type of vanVleck correction for spectrometer data.

Synopsis
setvv(vv)

Arguments

vv	Valid types are “schwab”, “old”, “none”, and “default” Allowed:string

Returns
boolean

Description
The type of vanVleck correction is only relevent to data from the spectrometer. The “schwab” correction is an implementation of a general solution developed by Fred Schwab. The “old” correction is the original vanVleck correction based on code from Arecibo. If “none” is selected, no vanVleck correction will be done. The current default, if this is not set, is “Schwab”. If the vv argument value is unrecognized, the return value will be F (false) and the default type will be used. This argument is case insensitive.

---

gbtmsfiller.vv - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the value of type of vanVleck correction being used.

Synopsis
vv()

Returns
string

---

gbtmsfiller.setsmooth - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Choose the type of smoothing for spectrometer data.

Synopsis
setsmooth(vv)

Arguments

vv	Valid types are “hanning”, “hamming”, “none”, and “default” Allowed:string

Returns
boolean

Description
The type of smoothing is only relevent to data from the spectrometer. This smoothing is applied after the vanVleck correction is done and before the data are fourier transformed to the spectral domain. The current default, if this is

not set, is "hanning". If the smooth argument value is unrecognized, the return value will be F (false) and the default type will be used. This argument is case insensisitive.

---

gbtmsfiller.smooth - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the value of type of smoothing being used.

Synopsis
smooth()

Returns
string

---

gbtmsfiller.setusehighcal - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Set the flag which controlls which CAL_TEMP is used from the receiver cal table.

Synopsis
setusehighcal(usehighcal)

Arguments

usehighcal	When true, HIGH_CAL_TEMP is used, else LOW_CAL_TEMP. Allowed:boolean Default: F

Returns
boolean

Description

The receiver cal tables contain measured TCAL values for a HIGH and LOW cal as a function of frequency for each receptor. In early GBT data, the IF FITS file lacked the information to determine which cal was appropriate. This flag is used to signal, for that old data, which cal value to use. It is ignored for current data since the IF fits file fully specifies which TCAL value is used. When this flag is true and the IF FITS file is missing this information, the HIGH cal value is used.

---

gbtmsfiller.usehighcal - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the value of the usehighcal flag.

Synopsis

usehighcal()

Returns

boolean

---

gbtmsfiller.setusebias - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Set the flag which controlls whether the bias is calculated during the van Vleck correction.

Synopsis

setusebias(usebias)

Arguments

usebias	When true, the bias is part of the vanVleck correction for ACS data. Allowed:boolean Default: F

Returns
boolean

Description
There is often a non-zero bias in the ACS (spectrometer) sampler levels. When this flag is true, the van Vleck correction attempts to work out what this bias is by examining the last 5 percent of the lag values.

---

gbtmsfiller.usebias - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the value of the usebias flag.

Synopsis
usebias()

Returns
boolean

---

gbtmsfiller.setoneacsms - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Set the flag which controlls whether a single ACS MeasurementSet is filled.

Synopsis
setoneacsms(oneacsms)

Arguments

oneacsms	When true, a single ACS MeasurementSet is filled with data from all ACS banks. Allowed:boolean Default: T

Returns
boolean

Description
The original filler filled each ACS bank to a separate MeasurementSet. This makes bulk processing of all ACS data (from all banks) difficult so the filler was rewritten to allow it to put all ACS data, from any bank, to the same MeasurementSet. That is now the default behavior. There may be circumstances where it is desirable to separate them out into separate MeasurementSets and this flag allows you to do that.

---

gbtmsfiller.oneacsms - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the value of the oneacsms flag.

Synopsis
oneacsms()

Returns
boolean

---

gbtmsfiller.setcompresscalcols - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Set the flag which controls whether the calibration columns are compressed.

Synopsis
setcompresscalcols(compresscalcols)

Arguments

compresscalcols	When true, the calibration columns are compressed. Allowed:boolean Default: T

Returns
boolean

Description
In order to control data bloat, it is desirable to compress the calibration columns (e.g. COMPRESSED_DATA). The compression is done independently for each row. This only applies to spectral line data (DCR and Holography data are never compressed). When this flag is true, these columns are compressed.

---

gbtmsfiller.compresscalcols - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the value of the compresscalcols flag.

Synopsis
compresscalcols()

Returns
boolean

---

gbtmsfiller.setdcbias - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Set a specific bias to use during the van Vleck correction.

Synopsis
setdcbias(dcbias)

Arguments

dcbias	This value is used as the bias. Allowed:double Default: 0.0

Returns
boolean

Description
There is often a non-zero bias in the ACS (spectrometer) sampler levels. If the bias is thought to be known based on other scans, then it can be set here to a specific value. This is the actual bias in the samplers, and not simply the observed asymptote in the raw lags. If uselag is True, then this value is ignored in favor of the derived dcbias.

---

gbtmsfiller.dcbias - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the specific value of the dcbias to use during the van Vleck correction.

Synopsis
dcbias()

Returns
double

---

gbtmsfiller.setminbiasfactor - Function

Package	**nrao**

Module	**gbt**
Tool	**gbtmsfiller**

Attempt to correct for the truncation error in the ACS samplers.

Synopsis
setminbiasfactor(minbiasfactor)

Arguments

minbiasfactor	The minimum integer to be multiplied by 0.5/65536 and subtracted from the data.. Allowed:integer Default: -1

Returns
boolean

Description
There is a truncation (instead of a rounding) as the raw lags are converted to 32-bit integers. This introduces a systematic error in the lags. When this value is non-negative (¿=0) then the nearest integer larger than this value is found which, when multiplied by 0.5/65536 and added to the last 5to just above 0.0.

---

gbtmsfiller.minbiasfactor - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the minbiasfactor value.

Synopsis
minbiasfactor()

Returns
integer

---

gbtmsfiller.newms - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Force a new MS for all currently filling backends.

Synopsis
newms()

Returns
boolean

Description
This is not yet implemented.

---

gbtmsfiller.setfixlags - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Toggle whether bad ACS lags should be fixed, if possible.

Synopsis
setfixlags(fixlags)

Arguments

fixlags	When true, bad ACS lags are fixed, if possible. Allowed:boolean Default: False

Returns
boolean

Description

The ACS some times suffers from bad 1024-lag segments where the entire segment is obviously offset and possibly scaled differently from the surrounding lags. If this toggle is true and that bad-lag segment is not the first 1024 lags, then the filler will "fix" the bad lags. This fix is done by first subtracting the median of those bad lags from the bad lags. Then the bad lags are scaled so that their standard deviation matches the standard deviation of the 512 lags on either side of the bad lags. Finally, a constant is added to the bad lags so that the median of the bad lags matches the median of the 512 lags on either side of the bad lags. If this toggle is false, the data are flagged as bad.

---

gbtmsfiller.fixlags - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Get the fixlags toggle value.

Synopsis
fixlags()

Returns
boolean

---

gbtmsfiller.done - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Stop filling and close the tool.

Synopsis
done()

Returns
boolean

Description

This permantly closes the connection between this `tool` and any data being filled. Use this only when you are done with the current `tool`. Once this function has been called it becomes useless. Any MeasurementSets and data files currently opened when this function is called will be closed.

---

gbtmsfiller.type - Function

Package	**nrao**
Module	**gbt**
Tool	**gbtmsfiller**

Return the type of this tool.

Synopsis
type()

Returns
string

Description
This is required for the toolmanager to manage this tool. This function always returns "gbtmsfiller".

1.1.3 gbtlogfiller - Function

Package	**nrao**
Module	**gbt**

Join GBT logs into a single AIPS++Table

Synopsis
gbtlogfiller()

Description
This is a stand-alone program run from the unix prompt that joins GBT log data into a single AIPS++table.

Usage: gbtlogfiller startTime endTime outTable dir1 dir2 ...
startTime and endTime are expressed as : mm/dd/yyyy,hh:mm:ss
outTable is the output name of the resulting table
dir1, dir2, ... are the names of the log directories to be examined and potentially joined into the output table.

1.1.4 gbtlogview - Function

Package nrao
Module gbt

View GBT logs

Synopsis
gbtlogview()

Description
This Glish function allows viewing of GBT logs after filling into an AIPS++ table using gbtlogfiller.
This can be run as a stand-alone program by typing 'gbtlogview2' or 'gbtlogview3' from the command line for the 'new' or 'test' versions of the program.
The program can also be run as a Glish client; typing 'include "gbt.g"' begins execution and fires up the "GBT Log Data" GUI.
For more detailed information see Note 207.

1.2 vla - Module

Module for VLA specific data processing
include "vla.g"

Description

This module contains functions and tools that are targeted at processing data from the Very Large Array (VLA). AIPS++ includes many other functions and tools for processing VLA data. These functions are generic enough to process data from other telescopes and hence are not described here. The functions and tools described below are specific to the VLA.

Tools

vlafiller	A tool for converting VLA archive format data to a measurement set

Functions

vlafillerdemo	Demonstrate the vlafiller tool
vlafillerfromdisk	Convert VLA archive data from a disk file to a measurement set
vlafillerfromtape	Convert VLA archive data from a tape to a measurement set
vlafilleroldweights	Use an older weighting scheme
vlafillertest	Test the vlafiller tool

1.2.1 vlafiller - Tool

Package **nrao**
Module **vla**

A tool for converting VLA archive format data to a measurement set
include "vlafiller.g"

Description
A vlafiller is a tool that contains functions that allow you to convert data that has been stored in the VLA archive format to a measurement set. Note that VLA archive format versions earlier than version 23 (earlier than 1990) are not supported by the vlafiller.

The VLA archive format is radically different from FITS and is the native data format written by the online computers at the VLA. The VLA data that is archived at the AOC is stored, on tape, in this format. It is described more fully in the VLA Computer Memorandum 188.

Running the vlafiller tool is a multi-step process. These steps can be grouped into the following sequence.

1. Create a vlafiller tool
2. Specify the data source and sink
3. Select some data filters (optional)
4. Start the copying process
5. Remove the tool

The advantage of a multi-step approach is that it gives the user fine grained, flexible control of the filling process. For many users this is unnecessary and the global functions vlafillerfromtape, vlafillerfromdisk bundle the vlafiller functions into a simpler one step approach.

Creating a vlafiller tool
The only way to create a vlafiller tool is to use the vlafiller constructor. Calling this function will start a precompiled executable, called vlafiller, usually running on your machine. This executable executes all the vlafiller functions. See the documentation for the vlafiller constructor for a more complete description of this function.

Specifying the data source and sink
The data source can either be a tape drive, disk file or the online computers. If it is a tape drive then it is assumed that a VLA archive format tape is already in the drive. The vlafiller knows about the layout and blocking structure of files on a VLA archive format tape and is able to rewind and skip to the any file on the tape. The files on a tape are numbered beginning at one. Users of tapes from the VLA archive need to be aware that the first file on these tapes is a tape label and hence the first file containing astronomical data will be the file numbered two.
An alternative to using the tapeinput function the VLA archive data may be copied from tape to disk using commands like:

```
dd if=/dev/tape bs=26624 of=file1.vla
```

and you can use commands like

```
mt -f /dev/tape fsf 2
```

to manually skip, in this example, over two tape files.
Then the diskinput function can be used to read this file. The advantage of this is that reading from disk is *much* faster and a lot less troublesome than reading from tape. It is also advantageous to do this if you need to read the data multiple times as shown in the examples below. The disadvantage is that

you need about 160MBytes of spare disk space, as this is typically the size of a VLA archive file.
The vlafiller can also read data directly from the online computers at the VLA. This option is only available when the filler is run on a computer at the array operations center in Socorro. See the final example below for an illustration of how to do this.
The data sink is a measurement set. The vlafiller can append data to an existing measurement set[1] or create a new one. Only one measurement set can be written for each pass through the input data. If you need to write multiple measurement sets each containing a different subset of the input archive you should consider copying the data to disk as described above. See the documentation for the output function for a more complete description of how to specify the data sink.

Selecting data filters
One measurement set is able to contain all the data in any number of VLA archive format files. Such a measurement set will usually be quite large, potentially affecting data access times, and may contain unwanted data. The vlafiller has a number of functions that allow you to discard data prior to it being copied into the output measurement set. These functions all have names that begin with the word select, a typical one being selectproject.
It is possible to use any combination of select functions as each one filters the data on a different parameter[2]. For example using the selectproject and selectfrequency functions will then only copy data from the specified project *and* in the specified frequency range. Other data in the same project or other projects that use the same frequency will not have their data copied to the measurement set.
Using a specific select function more than once will overwrite the previous selection. For example using the selectsource function with sourceA and then sourceB as arguments will discard any selection of sourceA (assuming you have not run the fill function between the two function calls).
Currently it is only possible to select data based on a limited number of parameters. Other selections must be done after the data is converted to a measurement set.

Starting the copying process
Once a vlafiller tool has been setup with input and output files the fill function is used to copy the data.
This function, unlike all the other ones in the vlafiller tool, can take some time. Because of this it can be run in the background ie., asynchronously. If run synchronously and from the command line you will not be able to type any further commands to glish until the command completes. If run asynchronously from the command line the function will return with a small

[1]Currently, it is only possible to append to a measurement set that was created with the vlafiller tool.
[2]The exception is the selectfrequency and selectband functions which are mutually exclusive.

positive integer that indicates the job number. This number can be used to query whether the function is still running and a message will appear in the logger when the function has finished. In either case the vlafiller tool will not execute any other functions until the fill function has completed. You need to start a new vlafiller server process, by calling the constructor with the forcenewserver argument set to true, to run two fill functions simultaneously. You can signal for the filler to stop filling as soon as possible using the stop function. This function will shut the filling function down after it has completed processing the current record.
When the fill function terminates it closes the input source, but leaves the output measurement set open. This way you can open another input source and append data to the existing measurement set.

Removing the tool
When you have finished with the vlafiller you should shut down the process, using the done function, even though it is not consuming much processor time. This is to conserve virtual memory. Once a tool is shut down it can no longer be used. It also closes the output measurement set.

Example
The following example shows the minimum necessary to run the vlafiller. The first line makes all the definitions for this tool available to the user. The second line calls the constructor. The third tells the filler that the input is to come from a file, in the current directory, called file1.vla. The fourth states that the output is to go to a measurement set called file1.ms. The fifth line does all the copying and the sixth shuts down the tool.

```
include 'vlafiller.g'     #1
v := vlafiller()          #2
v.diskinput('file1.vla')  #3
v.output('file1.ms')      #4
v.fill()                  #5
v.done()                  #5
```

This example could also be done much more simply using the vlafillerfromdisk global function as shown below.

```
vlafillerfromdisk('file1.ms', 'file1.vla')
```

All the other examples in this section are too complicated to be done using the vlafillerfromdisk or vlafillerfromtape global functions.

Example
This example illustrates the way the vlafiller can be used to read all the calibrators from three files.

The first three lines are identical to the first example. The fourth line sets the output measurement set to file1.ms. As this file may already exist, if you have just done example shown above the overwrite flag is set to force deletion of any existing file prior to copying any data. The fifth line selects all the calibrators, irrespective of their calibration code. The sixth line copies all the calibrators from the file to the measurement set.
At this point the input VLA archive file is closed but the output measurement set is still open. Line seven opens a new VLA archive file, called file2.vla, and line eight copies all the calibrators from that file to the measurement set. Note that the selection done in line five is still current and that the second fill command appends to the existing measurement set rather than overwriting it. This process continues for the third file before the process is shut down.

```
include 'vlafiller.g'              #1
v := vlafiller()                   #2
v.diskinput('file1.vla')           #3
v.output('file1.ms', overwrite=T)  #4
v.selectcalibrator('*')            #5
v.fill()                           #6
v.diskinput('file2.vla')           #7
v.fill()                           #8
v.diskinput('file3.vla')           #9
v.fill()                           #10
v.done()                           #11
```

Example
This example illustrates how to make multiple passes through a file to do more complex selections. The selectsource function can only select one source. If you want to create a measurement set with exactly two sources you need to read the file twice as shown in the example below.
The first four lines are identical to the above example. The fifth line selects the first source 'ngc1234' and the sixth line copies this to the measurement set. As described above after calling the fill function the input data source is closed. So we need to re-specify it as shown in line seven. Line eight then sets the selection to read the other source and line nine appends it to the measurement set.

```
include 'vlafiller.g'              #1
v := vlafiller()                   #2
v.diskinput('file1.vla')           #3
v.output('file1.ms', overwrite=T)  #4
v.selectsource('ngc1234')          #5
v.fill()                           #6
v.diskinput('file1.vla')           #7
```

```
v.selectsource('cen-a')            #8
v.fill()                           #9
v.done()                           #10
```

Example
This example illustrates how to use the vlafiller to read from a tape on a remote machine and fill the data asynchronously (as tapes take a long time). The first line is identical to previous examples. The second line starts up the vlafiller process on the computer called 'tapehost' (thats the one with the tape drive). The third line tells the vlafiller it is to read the first three files from the tape in the drive with a Unix device name of '/dev/nst0'. The fourth line tells the vlafiller to create the measurement set in the /tmp directory. This is the /tmp directory on the tapehost machine which is usually not the same as the /tmp directory on your local machine. To see the measurement set on your local machine you need to write to a disk that shares files (usually via NFS). The fifth line selects only the data from the specified project. The sixth line tells the filler to start copying the data asynchronously.
At this point glish returns before all the data has been copied. When all the data has been copied a message appears in the logger window. This may be missed if you are doing other things so in this example I set up a whenever that detects the event signalling that the job has finished and pops up a window to attract the users attention. Finally the tool is shut down.

```
include 'vlafiller.g'                       #1
v := vlafiller(host='tapehost')             #2
v.tapeinput('/dev/nst0', files=[1,2,3])     #3
v.output('/tmp/file1.ms')                   #4
v.selectproject('ab123')                    #5
jobid := v.fill(async=T, verbose=F)         #6
whenever defaultservers.alerter() -> [as_string(jobid)] do {
  deactivate
  messagebox("VLA filler has finished")
}
....
v.done()
```

Example
The final example illustrates how to use the vlafiller to read data from the online computers.
The first two lines are identical to previous examples. The third line tells the vlafiller it is to read data from the online computers. The fourth line tells the vlafiller to create the measurement set in the current directory. The fifth line

selects only the data from the specified project. The sixth line tells the filler to start copying the data. Records are processed as they are made available by the online computers. To stop the filling process the stop function is used. This cleanly terminates the filling function and the tool is shutdown using the done function.

```
include 'vlafiller.g'                          #1
v := vlafiller()                               #2
v.onlineinput()                                #3
v.output('realtime.ms')                        #4
v.selectproject('ab123')                       #5
v.fill(async=T)                                #6
....
v.stop()                                       #7
v.done()                                       #8
```

Scaling & weighting of VLA data
The vlafiller scales both the visibility data and computes the weights in a manner that is compatible with the 31DEC01 version of the AIPS task FILLM. The details of both the scaling and weight computation are described below.
The visibility data that is stored in the VLA archive is related to the the raw correlator coefficients by a scaling factor. This factor, which is different for each correlator, can be decomposed into antenna based terms called the nominal sensitivity (s_i). The raw correlation coeffecient (ρ_{ij}) is then related to the value stored in the VLA archive (v_{ij}) by the formula:

$$v_{ij} = \rho_{ij} \cdot \sqrt{s_i \cdot s_j}$$

The nominal sensitivity is derived during the course of an observation from calibrated noise diodes on each antenna.
The visibility data in VLA archive format is represented using a 16-bit signed integer for each of the real and imaginary components. To increase the dynamic range that can be represented each group of visibilities is scaled by a signed exponent (e). A group of visibilities is, in continuum mode, all four-correlations ie., RR, LL, RL, LR, for one baseline. In spectral line mode it's all channels corresponding to one correlation.
In order to convert these integers to a complex number (d_r, d_i) the vlafiller uses the formula (the subscripts r & i denote the real and imaginary components):

$$(d_r, d_i) = (v_r, v_i)/2^{-(e+8)}$$

All floating point operations are done in single precision. The factor of 8 added to the exponent results in scaling the complex data by 256. It is probably used to further scale the data so that it has nominal units of deka-Jansky's ie., 1.0 means 10 Jy.

The vlafiller assigns weights to the data that are directly proportional to both the integration time and the observed bandwidth and indirectly related to the system temperature at each of the antennas. The mathematical description and justification of this is described in AIPS memo 103, "Weighting data in AIPS" by Ketan Desai. In essence it states that the weights should be given by:

$$w_{ij} = \frac{\Delta t \Delta \nu}{s_i s_j}$$

where: Δt is the integration time (typically 10 seconds)
$\Delta \nu$ is the bandwidth (often 50MHz)
s_i is the nominal sensitivity of antenna i (typically around .2)

AIPS modifies these weights by scaling them by a factor that has been empirically determined so that the rms determined from the weights ($\sigma_{ij} = \frac{1}{\sqrt{w_{ij}}}$) corresponds to the measured rms in the visiblities when a calibrator is being observed. This factor is .12/10000[3]. In order to maintain consistency with AIPS, AIPS++ also scales the weights by this factor.
Prior to version 1.5 of AIPS++ the vlafiller used a simpler algorithm for calculating weights that matched what was the default in older versions of AIPS. The old weights were simply the integration time, in seconds, divided by 10. The vlafiller contains a global function called vlafilleroldweights that allows you to change the weights in a measurement set so that the older scheme is used. You cannot change the weights in a measurement from the old scheme to the new scheme. Instead you need to reload the data.

Constructors

vlafiller	Construct an vlafiller tool.

Functions

diskinput	Specify the input data file.
done	Delete the vlafiller tool
fill	Copy the data
onlineinput	Read input data from the online computers.
output	Specify the output measurement set
selectband	Select which frequency-band to convert
selectcalibrator	Select which calibrators to convert
selectfrequency	Select which frequency-range to convert
selectproject	Select which project to convert
selectsource	Select which source to convert
selectsubarray	Select which sub-array to convert
selecttime	Select which time-range to convert

[3]It is interesting to note that .12/10000 is very close to $1/256^2$ which is the adjustment that would be made in the weights that corresponded to the scaling the data 256.

state	Show the state of the tool
stop	Stop the filling process
tapeinput	Specify the input tape device.

vlafiller - Constructor

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Construct an vlafiller tool.

Synopsis
vlafiller(host, forcenewserver)

Description

Use this constructor to create a vlafiller tool. This is the only way to construct such a tool. As described above newly created tools cannot convert data until the the input data source is defined, using the tapeinput, diskinput or onlineinput functions, and output measurement set is defined using the output function.
The host argument specifies which machine the precompiled vlafiller process should be started on. The default value of unset starts it on the same machine as the one that glish is running on. Starting a vlafilller process on a remote machine gives you access to the tape drives and disks on that machine. Use this option if you do not have a tape drive on your local machine.
In order to run the vlafiller on a remote machine you need to satisfy all the following conditions.

- It must be possible to start AIPS++ on the remote machine
- You must be able to log onto the remote machine without having to type a password
- The AIPSPATH environment variable must be defined on the remote machine. You may want to set this up in the relevant "dot" file eg., adding a line like `source /usr/local/aips++/aipsinit.csh` in your .cshrc file (for csh).

One quick way to check if all three conditions are met is to type, on your local machine:

```
rsh host 'echo \$AIPSPATH'
```

where host is replaced by the name of the remote machine. If the value of the AIPSPATH variabl that is printed does not contain something like `aips-root architecture site host` and that all the values are correct for the remote machine you can be certain that starting the vlafiller, or any AIPS++ server, on the remote host will not work

Each vlafiller server process can only run one function at a time. Hence it is not possible, for example, to simultaneously read data from two tape drives to two measurement sets using one server. To solve this you start two servers. The forcenewserver argument allows you to do this by overriding the default behaviour of having each vlafiller tool share the same server.

Arguments

host	The host on which the vlafiller should be run Allowed:string Default: unset
forcenewserver	Always start a new server Allowed:Boolean Default: F

Returns
A vlafiller tool

Example

```
include 'vlafiller.g'
v := vlafiller()
v.diskinput('file1.vla')
v.output('file1.ms')
v.selectproject(project='ab123')
v.state()
v.fill(async=T)
v.stop()
v.done()
```

This example creates a vlafiller tool, sets the data source as a disk file called file1.vla, and the data sink as the measurement set called file1.ms. The selectproject function filters the data so that only records that correspond to the case insensitive project-id 'ab123' will be copied. The state function prints the current state of the filler in the logger, in case you forget.

No data is actually copied until the fill function is called. If a large amount of data is involved then the fill function may take some time so it may make sense to let the filler work in the background by setting the async flag. After copying some data the user stopped the filling function using the stop function. The vlafiller tool was then shut down, memory associated with it released and the output measurement set closed using the done function.

---

vlafiller.diskinput - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Specify the input data file.

Synopsis
diskinput(filename)

Description

The diskinput functions specifies that the input data is the specified file on disk.
Normally VLA archive data is stored on tape but if you need to read one or more files repeatedly it may be faster to copy the file to disk and then convert this disk file to a measurement set. Data can be read from tapes to disk in a variety of ways including:

```
dd if=/dev/tape bs=26624 of=file1.vla
```

This functions returns F if the specified file is not readable.

Arguments

filename	The input file name. Allowed:string Default: no default

Returns
T or F

Example
See the example for the vlafiller constructor.

---

vlafiller.onlineinput - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Read input data from the online computers.

Synopsis
onlineinput()

Description

The onlineinput functions specifies that the input data is to come from the online computers at the VLA. This allows you to read data shortly after it has been observed. The data is sent from the VLA computers to a predefined machine called `vis-server` at the Array Operations Center (AOC) in Socorro NM. The vlafiller will make a connection, via the Internet to this machine. Online filling is only available on the computers at the AOC.

Returns
T or fail

Example
See the final example in the tool overview documentation.

---

vlafiller.tapeinput - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Specify the input tape device.

Synopsis
tapeinput(device, files)

Description

The tapeinput function specifies that the input data source is the specified files on the specified tape device.

VLA archive data is normally stored on tape. Each tape may contain numerous files with the data from different projects mixed within one file and/or spread across numerous tape files. The tape archive log will contain a summary of what data is on what tape file.

This function tells the vlafiller tool which tape device and which files to read from the specified tape device. The tape device is the standard Unix name. Typical examples are `/dev/rmt/0ln` for Solaris 2.x and `/dev/nst0` for Linux.

The files are numbered beginning at one. The files that are to be read are specified with positive integers that are in ascending order. For example specifying `files=[1,2,10]` will read the first, second and tenth files from tape. [4]

Users of AIPS should note that the AIPS++ convention of explicitly specifying which files to read is different, and more flexible, than than using the `NFILES` adverb in the `FILLM` task. In `FILLM` the `NFILES` adverb specifies how many tape files to skip prior to reading to reading any data.

The vlafiller tool always rewinds the tape in order to know where the the tape is currently positioned when the files are specified using integers greater than zero. If you have manually positioned the tape to the start of the file you want to read you can specify that the filler should "read the next file" using `files=0`. The number zero in this case meaning "read the next file". In this case the filler will not rewind the tape prior to reading the data. It is also possible to specify `files=[0,0,0]` which means "read the next three files" and to mix zeros with absolute file numbers. For example `files=[1,0,10]` will read the first, second and tenth files, but `files=[1,0,2]` will produce an error message as the filler will not read file two twice.

This functions returns F if the specified tape is not readable.

Arguments

device	The tape device. Allowed:string Default: no default
files	The files to read. Allowed:vector of integers Default: [1]

Returns

T or F

[4] Users should be aware that, for tapes in the VLA archive, this numbering scheme differs by one from that used in versions of aips++ prior to 1.7. The numbering scheme now corresponds to that listed in the logs kept in the archive room.

Example

```
include 'vlafiller.g'
v := vlafiller()
v.tapeinput('/dev/nst0', files=1)
v.output('file1.ms')
v.selectsource(source='ngc1234', qualifier=1)
v.fill()
v.done()
```

This example is identical to that for the vlafiller constructor except that the input data is the first file on the specified tape and all the data of ngc1234 with qualifier will be copied.

---

vlafiller.output - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Specify the output measurement set

Synopsis
output(msname, overwrite)

Description

The output function specifies that the name of the output measurement set. A measurement set stores the data on disk in files & directories within a directory with the specified name.
If the specified measurementment set already exists, and contains exclusively data that has been created by the vlafiller tool, it is possible to append new data to the existing measurement set. The overwrite argument needs to be set to F to append data.
Otherwise the measurement set, or a file/directory with the same name, will not be replaced unless the overwrite argument is set to T.
This functions returns F if the specified measurement set cannot be created.

Arguments

msname	The measurement set name. Allowed:string Default: no default
overwrite	Overwrite an existing file. Allowed:Boolean Default: F

Returns
T or F

Example
See the example for the vlafiller constructor.

---

vlafiller.selectproject - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Select which project to convert

Synopsis
selectproject(project)

Description

The selectproject function filters the data so that only some of it is copied to the output measurement set.
The observing project identifier is a string of six or less characters. Only records that contain the same, case in-sensitive, string are copied to the output measurement set. The 'unset' value matches all projects. This is the default.
This functions returns F if the selection could not be done.

Arguments

project	The observing project name. Allowed:string Default: unset

Returns
T or F

Example
See the example for the vlafiller constructor.

---

vlafiller.selecttime - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Select which time-range to convert

Synopsis
selecttime(start, stop)

Description

The selecttime function filters the data so that only some of it is copied to the output measurement set.
Only data that have been observed[5] between the specified start and stop times will be copied to the output measurement set. The times are specified using any of the string formats that can be interpreted by the quanta module (see the section on time formatting).
If the start input parameter is unset then it defaults to 'a long time in the past'. Similarly if the stop input parameter is unset it defaults to 'along time in the future'. Hence using this function with no input parameters disables any time selection.
It is important to specify both the day and time portions for the start and stop time. Otherwise the day will default to the day with the modified Julian day of zero ie. the 17-Nov-1858. If you want to fill in the current date for the day number you can preceed the time with a slash. eg., `start='/6:45'`.
This functions returns F if the selection could not be done.

Arguments

[5]The observation time of a single integration is the time at the midpoint of the integration

start	The beginning of the time range. Allowed:string Default: unset
stop	The end of the time range. Allowed:string Default: unset

Returns
T or F

Example

```
include 'vlafiller.g'
v := vlafiller()
v.diskinput('file1.vla')
v.output('file1.ms')
v.selecttime(start='31Dec00/00:00:00.123',
     stop='31-Dec-2000/23:59:59.876')
v.selectfrequency(centerfrequency='1.4GHz', bandwidth='200MHz')
v.fill()
v.done()
```

This example selects all the data that was observed on the last day of the millennium where any part of the observation was between 1.3GHz and 1.5GHz. See the example for the vlafiller constructor for a more complete discreption of other parts of this example.

---

vlafiller.selectfrequency - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Select which frequency-range to convert

Synopsis
selectfrequency(centerfrequency, bandwidth)

Description

The selectfrequency function filters the data so that only some of it is copied to the output measurement set.
The frequency band is specified using a center frequency and a bandwidth around that frequency. These frequencies are specified using strings to include the necessary units ie., kHz, MHz etc.
This function rejects input records were all the correlations are entirely outside the specified frequency band. It cannot be used to select channels, this must be done after the measurement set has been created. For example if a spectral line observation has data in ten channels between 100MHz and 200MHz then calling this function with a center frequency of 200MHz and a bandwidth of 40MHz will include *all* of the data between 100MHz and 200MHz and not just the last two channels.
If any of the input parameters is unset then no frequency selection is done. This is the default.
This functions returns F if the selection could not be done.

Arguments

centerfrequency	The center frequency. Allowed:string Default: unset
bandwidth	The range of frequencies around the center frequency. Allowed:string Default: unset

Returns
T or F

Example
See the example for the selecttime function.

---

vlafiller.selectband - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Select which frequency-band to convert

Synopsis
selectband(bandname)

Description

The selectband function filters the data so that only some of it is copied to the output measurement set.
This function selects input records where any of the correlations are within one of the eight standard VLA observing bands. These bands are:

Band	Frequencies
4	48 – 96 MHz
P	298 – 345 MHz
L	1.15 – 1.75 GHz
C	4.2 – 5.1 GHz
X	6.8 – 9.6 GHz
U	13.5 – 16.3 GHz
K	20.8 – 25.8 GHz
Q	38.0 – 51.0 GHz

The input argument to this function is a single character, case insensitive, string containing one of the band codes shown above. This is then translated to the relevant frequencies and passed to the selectfrequency function described above. Like the selectfrequency function this function may also copy data that is outside the band if some of the data in an input record is inside the specified band.
If the input parameter is unset then no frequency selection is done. This is the default.
This functions returns F if the selection could not be done.

Arguments

bandname	The frequency band to select. Allowed:string Default: unset

Returns
T or F

Example
See the example for the selecttime function.

---

vlafiller.selectsource - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Select which source to convert

Synopsis
selectsource(source, qualifier)

Description

The selectsource function filters the data so that only some of it is copied to the output measurement set.
The source name is a string with up to 16 characters. Only records that contain the same, case in-sensitive, string are copied to the output measurement set. The 'unset' value matches all projects. This is the default.
The source qualifier is an additional parameter that can be used to distinguish different observing configurations of the same source. It is an integer and is usually zero. The unset value can be used to select data with any source qualifier and this is the default.
This functions returns F if the selection could not be done.

Arguments

source	The source name. Allowed:string Default: unset
qualifier	The qualifier. Allowed:integer Default: unset

Returns
T or F

Example
See the example for the tapeinput function.

---

vlafiller.selectsubarray - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Select which sub-array to convert

Synopsis
selectsubarray(subarray)

Description

The selectsubarray function filters the data so that only some of it is copied to the output measurement set.
The input argument to this function is an integer between one and five (inclusive) which indicates which subarray to select the data from.
If the input parameter is unset then no subarray selection is done. This is the default.
This functions returns F if the selection could not be done.

Arguments

subarray	The subarray id. Allowed:integer Default: unset

Returns
T or F

Example
See the selectcalibrator function.

---

vlafiller.selectcalibrator - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Select which calibrators to convert

Synopsis
selectcalibrator(calcode)

Description

The selectcalibrator function filters the data so that only some of it is copied to the output measurement set.
The calibrator code can be any uppercase letter or a single numeric character. Often it is assigned by the scheduling software and may have a specific meaning. The character `'*'` can be used to select all the calibrators regardless of the calibration code. This does not select any of the target sources which have a calibrator code of `' '` (the space character).
This function can be used to filter out all the calibrators so that only target sources are filled. This is done using a argument of `' '` (the space character).
To revert to the default behaviour of selecting all the target sources and all the calibrators this function should called with the unset argument.
This function returns F if the selection could not be done.

Arguments

calcode	The calibrator code to select. Allowed:string Default: unset

Returns
T or F

Example

```
include 'vlafiller.g'
v := vlafiller()
v.diskinput('file1.vla')
v.output('file1.ms')
v.selectband('L')
v.selectsubarray(1)
v.selectcalibrator('C')
v.fill()
v.done()
```

This example selects all the calibrator data with positions known to better than 50mas (as thats what calcode 'C' usually means), that was observed using in 'L' band (near 1.4GHz) in using sub-array 1. See the example for the vlafiller constructor for a more complete description of other parts of this example.

---

vlafiller.fill - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Copy the data

Synopsis
fill(verbose, async)

Description

The fill function copies the data from the previously specified input VLA archive to the previously specified output measurement set, applying any previously specified data filtering criteria.
All cross-correlation and auto-correlation data are copied to the measurement set. Data is copied and flagged if the online system indicates that any of the IF's involved in the correlation had a status greater than three.
Unlike most of the other functions in this tool the fill data may take a long time to execute. As a result it sends to the logger messages indicating whenever it has come across, a change in the array configuration, field center, spectral window or polarization setup. More extensive messages that log the processing of each record may be enabled by setting the verbose argument to T.
The fill function may also be run asynchronously by setting the async argument to T. When run asynchronously the function returns quickly and runs in the background. A message is sent to the logger window when the function has finished executing.
When the fill function has successfully completed the input source is reset to a null value. This means you *must* call setinput at least once once prior to every invocation of the fill function.
This functions returns T or the job id (if run asynchronously).

Arguments

verbose	Send messages when copying each record. Allowed:Boolean Default: F
async	Run this function asynchronously. Allowed:Boolean Default: F

Returns
T or a small integer

Example
See the example for the vlafiller constructor or the examples in the vlafiller tool description.

---

vlafiller.state - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Show the state of the tool

Synopsis
state()

Description

The state function displays to the logger the current state of the filler. This lets you know what the input and output files are and what selections have been made.

Returns
T or fail

Example
See the example for the vlafiller constructor.

---

vlafiller.stop - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Stop the filling process

Synopsis
stop()

Description

The stop function will signal the filler process to stop immediately after reading the next record. This is the earliest point at which the filling process can sensibly be halted and it is quite likely that the filling process will be stopped midway through a scan. Using this function, and then shutting down the filler process, using the done function, should result in a perfectly valid but truncated measurement set.

Returns
T or fail

Example
See the example for the vlafiller constructor.

---

vlafiller.done - Function

Package	**nrao**
Module	**vla**
Tool	**vlafiller**

Delete the vlafiller tool

Synopsis
done()

Description

The done function frees up all the memory associated with a vlafiller tool. This function also closes all the disk files and tape devices that it used so that other tools can use them. It will also shut down the server process if there are no other vlafiller tools being used.

Returns
T

Example
See the example for the vlafiller constructor.

1.2.2 vlafillerfromtape - Function

Package **nrao**
Module **vla**

Convert VLA archive data from a tape to a measurement set

Synopsis
vlafillerfromtape(msname, device, files, project, start, stop, bandname, source, overwrite, verbose, async, host)

Description

This global function prevents users from having to explicitly use the vlafiller tool by packaging its functions so they perform a common task ie., copying VLA archive data from tape to a measurement set.
There are only two arguments which the user must supply in order to successfully use this function, these are the device and msname arguments. These arguments specify the name of the tape device and the output measurement set. All other arguments are optional. See the tapeinput & output functions for more details.
A number of arguments control what data is copied to the output measurement set. These are:

project See the selectproject function.

start & stop See the selecttime function.

bandname See the selectband function.

source See the selectsource function[6].

The remaining arguments control details of the filling process. The overwrite argument specifies whether the output measurement set is deleted prior to copying the data. See the output function for more details.
The files argument controls which files on tape are to be read. The first file containing astronomical data is numbered file one and a number of files can be

[6] You cannot select which qualifier to use, you always get all of them

specified using a vector of increasing integers eg., `files=[1,2,10]`. See the tapeinput function for more details.

The verbose argument controls how much status information is sent to the logger and the async argument controls whether the function should run in the background. See the fill function for more details.

The host argument controls which machine this filler process is run on. This allows remote tape drives to be used. Unless you have done this before it is recommended you read the full description of this mechanism described in the vlafiller constructor.

This function returns T is everything went well and F otherwise.

Arguments

msname	The measurement set name. Allowed:string Default: no default
device	The tape device. Allowed:string Default: no default
files	The files to read. Allowed:vector of integers Default: [1]
project	The observing project name. Allowed:string Default: unset
start	The beginning of the time range. Allowed:string Default: unset
stop	The end of the time range. Allowed:string Default: unset
bandname	The frequency band to select. Allowed:string Default: unset
source	The source name. Allowed:string Default: unset
overwrite	Overwrite an existing file. Allowed:Boolean Default: F
verbose	Send messages when copying each record. Allowed:Boolean Default: F
async	Run this function asynchronously. Allowed:Boolean Default: T
host	The host on which the vlafiller should be run Allowed:string Default: unset

Returns
T or F

Example

```
include 'vlafiller.g'
```

```
vlafillerfromtape('ab123.ms', '/dev/rmt/0ln', files=[1,2], project='ab123');
```

This example creates the measurement set called 'ab123.ms' from all the records in the first two files of the tape that come from the observing project called 'AB123'.

1.2.3 vlafillerfromdisk - Function

Package nrao
Module vla

Convert VLA archive data from a disk file to a measurement set

Synopsis
vlafillerfromdisk(msname, filename, project, start, stop, bandname, source, overwrite, verbose, async, freqTolerance)

Description

This global function prevents users from having to explicitly use the vlafiller tool by packaging its functions so they perform a common task ie., copying data from a disk file containing VLA archive data into a measurement set.
In the best case, there are only two arguments which the user must supply in order to successfully use this function, these are the filename and msname arguments. These arguments specify the names of the input vla archive file and the output measurement set. All other arguments are optional. See the diskinput & output functions for more details. Due to the variety of the dataset quality, however, some dataset requires a larger tolerance for frequeny in order to recognize the same spectral window. So the user may need to pass in a proper tolerance according to the specific dataset. As a rule of thumb, executing the filling script first with the default tolerance and examing the generated SPECTRAL_WINDOW subtable of MS will give you a good hint about what is the right tolerance for the frequency to use. For instance, for dataset G192, the proper tolerance for frequency is about 6 times of its channel width(see the glish test script g192_testTolerance.g), while for dataset NGC7538, a much larger tolerance, 60 times of its channel width, (see the glish test scripts ngc7538_testTolerance.g) is required in order to recognize the two related frequencies as the same spectral window. The freqTolerance should be in the unit of Hz.
A number of arguments control what data is copied to the output measurement set. These are:

project See the selectproject function.

start & stop See the selecttime function.

bandname See the selectband function.

source See the selectsource function[7].

The remaining arguments control details of the filling process. The overwrite argument specifies whether the output measurement set is deleted prior to copying the data. See the output function for more details.

The verbose argument controls how much status information is sent to the logger and the async argument controls whether the function should run in the background. See the fill function for more details.

This function returns T is everything went well and F otherwise.

Arguments

[7] You cannot select which qualifier to use, you always get all of them

msname	The measurement set name. Allowed:string Default: no default
filename	The input file name. Allowed:string Default: no default
project	The observing project name. Allowed:string Default: unset
start	The beginning of the time range. Allowed:string Default: unset
stop	The end of the time range. Allowed:string Default: unset
bandname	The frequency band to select. Allowed:string Default: unset
source	The source name. Allowed:string Default: unset
overwrite	Overwrite an existing file. Allowed:Boolean Default: F
verbose	Send messages when copying each record. Allowed:Boolean Default: F
async	Run this function asynchronously. Allowed:Boolean Default: F
freqTolerance	The tolerance for frequency. Allowed:Double Default: The channel width of the current spectral window in Hz

Returns
T or F

Example

```
include 'vlafiller.g'
vlafillerfromdisk('ab123.ms', 'file1.vla', project='ab123');
```

This example creates the measurement set called 'ab123.ms' from all the records in the file called 'file1.vla' that come from the observing project called 'AB123'.

Example

```
include 'vlafiller.g'
vlafillerfromdisk(filename='data/G192/AS758\_C030425.xp1',
                  msname='g192\_a.ms', overwrite=T,bandname='K',
                  async=F, freqTolerance=150000.0);
```

This example shows how a user can pass in a tolerance for the frequency matching. It creates the measurement set called 'g192_a.ms' from all the records in the file called 'data/G192/AS758_C030425.xp1'.

1.2.4 vlafilleroldweights - Function

Package nrao
Module vla

Use an older weighting scheme

Synopsis
vlafilleroldweights(msname)

Description

This global function will change the weights of the supplied measurement set to a value that only depends on the integration time. The values in the WEIGHT column will be changed to one tenth of the value in the EXPOSURE column and the values in the SIGMA column will be changed to the inverse square-root of the new value in the WEIGHT column. This was the original behaviour of the vlafiller tool.
From release 1.5 the vlafiller tool sets the weights to a value that depends on the integration time, bandwidth and system temperature of the relevant antennas. This is believed to be more representative of the errors in the observation.
This function returns T is everything went well and F otherwise.

Arguments

msname	The measurement set name. Allowed:string Default: no default

Returns
T or F

Example

```
include 'vlafiller.g'
vlafilleroldweights('ab123.ms');
```

This example resets the weights in the measurement set 'ab123.ms'. There is no way to back out of this change so if you need to compare both weighting schemes you should copy your data to a separate measurement set prior to running this function.

1.2.5 vlafillerdemo - Function
Package nrao
Module vla

Demonstrate the vlafiller tool

Synopsis
vlafillerdemo()

Description

Returns
T or fail

Example

```
include 'vlafiller.g'
vlafillerdemo()
```

1.2.6 vlafillertest - Function

Package nrao
Module vla

Test the vlafiller tool

Synopsis
vlafillertest()

Description

Returns
T or fail

Example

```
include 'vlafiller.g'
vlafillertest()
```

1.3 e2e - Module

Module for End-to-end processing
include "e2e.g"

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use. The `e2e` module implements two inter-related services.

An archive A repository for observations from telescopes, and for pipeline-processed results.

A pipeline A facility for processing data from specific projects in the archive, and inserting processed results in the archive.

These services are designed to be used both from within AIPS++ and from the web.
An overview of the tools:

e2ecatalogs is responsible for constructing and accessing catalogs.

e2epipeline is responsible for constructing and executing scripts to process data.

e2epipelinequery is responsible for serving queries to e2epipeline.

e2emakefiles contains the various Glish makefiles that process data.

e2estandards is responsible for specifying standard values to be used in processing (*e.g.* cellsizes, image sizes, solution intervals, *etc.*).

e2epublish is responsible for publishing results to a standard location, as FITS files, and with information web pages.

e2emsutils has utility helper functions for MeasurementSet access.

e2eimagingutils has utility helper functions for imaging.

e2edisplayutils has utility helper functions for display.

e2epipelineutils has utility helper functions for the pipeline.

e2echromepipe connects glish to a web server.

This module uses a standard directory layout, rooted at the environmental variable E2EROOT. This allows various and mirrored copies of archives to be used. However, the subdirectory structure must be as expected.

archive Archive root

archive/catalogs Catalogs

archive/catalogs/TELESCOPE Catalogs for telescope TELESCOPE

archive/data Original (and untouched) data

archive/data/TELESCOPE Data for telescope TELESCOPE. This directory and below have telescope-dependent contents to reflect the differing nature of data from various telescopes.

archive/data/VLA/tapes Data tape images for the VLA.

archive/results Results from processing

archive/results/projects Results from processing for projects

archive/results/projects/PROJECT Results from processing for project PROJECT. All results must be indexed by a file PROJECT.html or PROJECT.xml.

archive/results/calibrators Calibrator source list

processing Processing area root. This has little substructure as yet.

Tools

e2ecatalogs	Tool for constructing catalogs for e2e processing
e2echromepipe	Connects glish to a web server
e2edisplayutils	Helper tool for e2e processing
e2eimagingutils	Helper tool for e2e processing
e2emakefiles	Tool for makefiles for e2e processing
e2emsutils	Helper tool for e2e processing
e2epipeline	Tool for pipeline processing
e2epipelineutils	Helper tool for e2e processing
e2epublish	Tool for publishing results of e2e processing
e2estandards	Tool for providing standards for e2e processing
mscatalog	Tool for constructing catalogs of MeasurementSets

1.3.1 mscatalog - Tool

Package	**nrao**
Module	**e2e**

Tool for constructing catalogs of MeasurementSets
include "mscatalog.g"

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.

Constructors

mscatalog	Construct an mscatalog tool

Functions

done	Done the tool
makeindex	Construct an index for the MeasurementSet
write	Write the catalog information as either SQL or an AIPS++ table

mscatalog - Constructor

Package	**nrao**
Module	**e2e**
Tool	**mscatalog**

Construct an mscatalog tool

Synopsis
mscatalog(msname, catalogname)

Description
This constructs an mscatalog for a given MeasurementSet. This returns a Glish variable containing the tool functions of mscatalog. The tools may then be used to construct an index and write out the results.

Arguments

msname	Name of MeasurementSet Allowed:String
catalogname	Name of catalog to be written Allowed:String Default: MSCATALOG

Returns
mscatalog tool

Example

```
m:=mscatalog('3C273XC1.MS')
m.makeindex()
m.write(/tables');
m.done();
```

---

mscatalog.makeindex - Function

Package	**nrao**
Module	**e2e**
Tool	**mscatalog**

Construct an index for the MeasurementSet

Synopsis
makeindex()

Description
The SCAN_NUMBER column is filled with a guess at scan numbers.

Returns
Bool

---

mscatalog.write - Function

Package	**nrao**
Module	**e2e**
Tool	**mscatalog**

Write the catalog information as either SQL or an AIPS++ table

Synopsis
write(what)

Description
The catalog may be written either as an AIPS++ Table or as a text file containging SQL statements.

Arguments

what	What to write? Allowed:String: tables—sql Default: tables	

Returns
Bool

---

mscatalog.done - Function

Package	**nrao**
Module	**e2e**
Tool	**mscatalog**

Done the tool

Synopsis
done()

Description
Finish using the tool. Use this to free up memory and resources when finished using the tool.

Returns
Bool

1.3.2 e2ecatalogs - Tool

Package nrao
Module e2e

Tool for constructing catalogs for e2e processing

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.

1.3.3 e2epipeline - Tool

Package nrao
Module e2e

Tool for pipeline processing
include "e2epipeline.g"

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.

Constructors

e2epipeline	Construct an e2epipeline tool

Functions

project	Process a project from the archive

e2epipeline - Constructor

Package nrao

Module	**e2e**
Tool	**e2epipeline**

Construct an e2epipeline tool

Synopsis
e2epipeline(e2edir)

Description
This is used to construct `e2epipeline` tools associated with a directory. If the directory is not specified, then the environmental variable E2EROOT is used. This returns a Glish variable containing the tool functions of e2epipeline.

Arguments

e2edir	Directory that is the root of the archive Allowed:String Default: F

Returns
e2epipeline tool

Example

```
e2ep:=e2epipeline('3C273XC1.MS')
e2ep.project('AB973', target='all')
```

---

e2epipeline.project - Function

Package	**nrao**
Module	**e2e**
Tool	**e2epipeline**

Process a project from the archive

Synopsis

project(project, telescope, tbeg, tend, execute, target)

Description
The data from a specified project are processed through a pipeline. The script is generated automatically from information in the archive, using an e2emakefiles.

Arguments

project	Name of project to process Allowed:String
telescope	Name of telescope from which the project originates Allowed:String Default: unset
tbeg	Beginning time Allowed:String Default: unset
tend	End time Allowed:String Default: unset
execute	Execute the script? Allowed:Bool Default: T
target	Name of makefile target Allowed:String Default: 'all'

Returns
Bool

1.3.4 e2emakefiles - Tool

Package nrao
Module e2e

Tool for makefiles for e2e processing

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.

1.3.5 e2estandards - Tool

Package	**nrao**
Module	**e2e**

Tool for providing standards for e2e processing
include "e2estandards.g"

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.

1.3.6 e2epublish - Tool

Package	**nrao**
Module	**e2e**

Tool for publishing results of e2e processing
include "e2epublish.g"

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.

1.3.7 e2edisplayutils - Tool

Package nrao
Module e2e

Helper tool for e2e processing

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.

1.3.8 e2eimagingutils - Tool

Package nrao
Module e2e

Helper tool for e2e processing

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.

1.3.9 e2emsutils - Tool

Package nrao
Module e2e

Helper tool for e2e processing

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.

1.3.10 e2epipelineutils - Tool

Package	**nrao**
Module	**e2e**

Helper tool for e2e processing

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.

1.3.11 e2echromepipe - Tool

Package	**nrao**
Module	**e2e**

Connects glish to a web server

Description

This is experimental and under rapid development: please do not use!
The NRAO End-to-End project (e2e) has the goal of providing end-to-end processing of data from telescopes. This is being developed for NRAO telescopes (VLA, VLBA, GBT, EVLA), but is being designed for general use.
`e2echromepipe` is used to connect an instance of glish to a web server.

Index