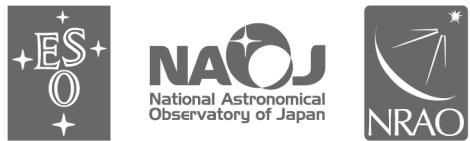
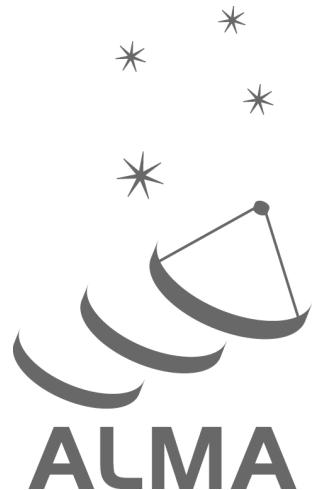


# ALMA Science Pipeline Reference Manual:

## CASA 4.5.1

### Interferometry and Single-Dish Data



[www.alma-science.org](http://www.alma-science.org)

---

ALMA, an international astronomy facility, is a partnership of ESO (representing its member states), NSF (USA) and NINS (Japan), together with NRC (Canada), NSC and ASIAA (Taiwan), and KASI (Republic of Korea), in cooperation with the Republic of Chile. The Joint ALMA Observatory is operated by ESO, AUI/NRAO and NAOJ.

## User Support:

For further information or to comment on this document, please contact your regional Helpdesk through the ALMA User Portal at [www.almascience.org](http://www.almascience.org). Helpdesk tickets will be directed to the appropriate ALMA Regional Center at ESO, NAOJ or NRAO.

## Revision History:

Version	Date	Editors
1.0	January 2016	Pipeline Team

## Contributors

Add names and logo's for any additional contributions beyond the ALMA Executives



In publications, please refer to this document as:

**ALMA Pipeline Team, 2016, ALMA Science Pipeline Reference Manual CASA 4.5.1, Version 1, ALMA**

# TABLE OF CONTENTS

<b>1</b>	<b>PURPOSE AND SCOPE .....</b>	<b>3</b>
<b>2</b>	<b>PIPELINE TASK TYPES .....</b>	<b>3</b>
<b>3</b>	<b>PIPELINE CONTEXT .....</b>	<b>4</b>
<b>4</b>	<b>PIPELINE TASK LIST .....</b>	<b>4</b>
<b>5</b>	<b>COMMON TASK DESCRIPTIONS .....</b>	<b>7</b>
5.1	h_init .....	7
5.2	h_resume .....	8
5.3	h_save .....	9
5.4	h_weblog.....	9
<b>6</b>	<b>INTERFEROMETRY COMMON TASK DESCRIPTIONS .....</b>	<b>10</b>
6.1	hif_antpos .....	10
6.2	hif_applycal .....	11
6.3	hif_atmflag .....	14
6.4	hif_bandpass.....	16
6.5	hif_bpflagchans.....	19
6.6	hif_cleanlist .....	21
6.7	hif_clean.....	23
6.8	hif_export_calstate .....	27
6.9	hif_exportdata .....	28
6.10	hif_findcont.....	31
6.11	hif_gaincal.....	32
6.12	hif_gainflag.....	35
6.13	hif_import_calstate .....	37
6.14	hif_importdata .....	37
6.15	hif_lowgainflag .....	39
6.16	hif_makelist .....	41

6.17	hif_makeimages .....	44
6.18	hif_makeimlist .....	46
6.19	hif_normflux .....	49
6.20	hif_rawflagchans .....	51
6.21	hif_refant .....	54
6.22	hif_restoredata .....	56
6.23	hif_setjy .....	58
6.24	hif_show_calstate .....	60
6.25	hif_tclean.....	61
<b>7</b>	<b>INTERFEROMETRY ALMA TASK DESCRIPTIONS .....</b>	<b>65</b>
7.1	hifa_antpos .....	65
7.2	hifa_bandpass.....	67
7.3	hifa_bpsolint.....	71
7.4	hifa_flagdata .....	74
7.5	hifa_fluxcalfag .....	77
7.6	hifa_fluxdb .....	79
7.7	hifa_gfluxscale .....	80
7.8	hifa_importdata .....	83
7.9	hifa_linpolcal .....	85
7.10	hifa_spwphaseup .....	87
7.11	hifa_timegaincal .....	90
7.12	hifa_tsyscal .....	93
7.13	hifa_tsysflagchans .....	94
7.14	hifa_tsysflagspectra .....	97
7.15	hifa_tsysflag .....	101
7.16	hifa_wvrgcalfag .....	104
7.17	hifa_wvrgcal .....	108
<b>8</b>	<b>SINGLE-DISH TASK DESCRIPTIONS .....</b>	<b>112</b>
8.1	hsd_applycal .....	112

8.2	hsd_baseline .....	114
8.3	hsd_blfloor .....	118
8.4	hsd_calsky .....	123
8.5	hsd_exportdata .....	126
8.6	hsd_flagdata .....	127
8.7	hsd_imaging .....	131
8.8	hsd_importdata .....	134
8.9	hsd_inspectdata .....	137
8.10	hsd_mstoscantable .....	138
8.11	hsd_plotflagbaseline .....	139
8.12	hsd_simplescale .....	141

## 1 Purpose and scope

The purpose of this document is to describe the tasks available for the calibration and imaging of interferometry datasets using the ALMA Science Pipeline.

## 2 Pipeline Task Types

There are 5 types of Pipeline tasks. This document provides descriptions of the task types: h\_, hif\_, hifa\_ and hsd\_.

**Table 1: Pipeline Task Types**

Task pre-fix	Task type	Description
h_	Common tasks	Pipeline tasks used in the calibration and imaging of both interferometry and single-dish datasets
hif_	Interferometry common tasks	Pipeline tasks used in the calibration and imaging of both ALMA and EVLA interferometry datasets
hifa_	Interferometry ALMA tasks	Pipeline tasks used in the calibration and imaging of ALMA interferometry datasets only
hifv_	Interferometry EVLA tasks	Pipeline tasks used in the calibration and imaging of EVLA interferometry datasets only

<b>hsd_</b>	Single-dish tasks	Pipeline tasks used in the calibration and imaging of single-dish datasets only
-------------	-------------------	---

### 3 Pipeline Context

The Pipeline state is stored in its context e.g. which calibration tables need to be used at each stage. Several Pipeline tasks are associated with initialising and editing (rarely needed) the context.

### 4 Pipeline Task List

Commissioned tasks are those used in the current ALMA standard recipes (`casa_pipescript.py` and `casa_piperestorescript.py`). Tasks which do not form part of the standard recipes are currently experimental. To assess which tasks were used in the processing of an ALMA dataset, please see the `casa_pipescript.py` and `casa_piperestorescript.py` included in ALMA deliveries for Pipeline-processed data.

**Table 2: Common Tasks**

	<b>Task name</b>	<b>Description</b>
1	<code>h_init</code>	Initialise the interferometry pipeline
2	<code>h_resume</code>	Restore a save pipeline state from disk
3	<code>h_save</code>	Save the pipeline state to disk
4	<code>h_weblog</code>	Open the pipeline weblog in a browser

**Table 3: Interferometry Common Tasks**

	<b>Task name</b>	<b>Description</b>
1	<code>hif_antpos</code>	Derive an antenna position calibration table
2	<code>hif_applycal</code>	Apply the calibration(s) to the data
3	<code>hif_atmflag</code>	Flag channels with bad atmospheric transmission
4	<code>hif_bandpass</code>	Compute bandpass calibration solutions
5	<code>hif_bpflagchans</code>	Flag deviant channels in bandpass calibration
6	<code>hif_cleanlist</code>	Compute clean map
7	<code>hif_clean</code>	Compute clean map
8	<code>hif_export_calstate</code>	Save the pipeline calibration state to disk
9	<code>hif_exportdata</code>	Prepare interferometry data for export

10	hif_findcont	Find continuum frequency ranges
11	hif_gaincal	Determine temporal gains from calibrator observations
12	hif_gainflag	Flag antennas with deviant gain
13	hif_import_calstate	Import a calibration state from disk
14	hif_importdata	Imports data into the interferometry pipeline
15	hif_lowgainflag	Flag antennas with low or high gain
16	hif_makelist	Compute list of clean images to be produced
17	hif_makeimages	Compute clean map
18	hif_makeimlist	Compute list of clean images to be produced
19	hif_normflux	Average calibrator fluxes across measurement sets
20	hif_rawflagchans	Flag deviant channels in raw data
21	hif_refant	Select the best reference antennas
22	hif_restoredata	Restore flagged and calibration interferometry data from a pipeline run
23	hif_setjy	Fill the model column with calibrated visibilities
24	hif_show_calstate	Show the current pipeline calibration state
25	hif_tclean	Compute clean map

**Table 4: Interferometry ALMA Tasks**

	<b>Task name</b>	<b>Description</b>
1	hifa_antpos	Derive an antenna position table
2	hifa_bandpass	Compute bandpass calibration solutions
3	hifa_bpsolint	Compute optimal bandpass calibration solution intervals
4	hifa_flagdata	Do basic flagging of a list of measurement sets
5	hifa_fluxcalflag	Locate line regions in solar system flux calibrator spws
6	hifa_fluxdb	Connect to flux calibrator database
7	hifa_gfluxscale	Derive flux density scales from standard calibrators
8	hifa_importdata	Imports data into the interferometry pipeline

9	hifa_linpolar	Compute polarization calibration
10	hifa_spwphaseup	Compute wide to narrow spectral window mapping and per spectral window phase offsets
11	hifa_timegaincal	Determine temporal gains from calibrator observations
12	hifa_tsyscal	Derive a Tsys calibration table
13	hifa_tsysflagchans	Flag deviant channels in system temperature measurements
14	hifa_tsysflagspectra	Flag deviant system temperature measurements
15	hifa_tsysflag	Flag deviant system temperature measurements
16	hifa_wvrgcalflag	Calculate WVR corrections
17	hifa_wvrgcal	Compute the WVR calibration

**Table 5: Single-Dish Tasks**

	<b>Task name</b>	<b>Description</b>
1	hsd_applycal	Apply calibration tables to the science target data
2	hsd_baseline	Detect and validate spectral lines, subtract a spectral baseline by masking detected lines
3	hsd_bflag	Flag spectra based on the pre-defined criteria of the single dish pipeline
4	hsd_calsky	Generate a calibration table from OFF source spectra
5	hsd_exportdata	Export the data products into the product directory
6	hsd_flagdata	Perform a-priori flagging of a list of measurement sets (MS)
7	hsd_imaging	Apply flag tables to the data, convert Scantables to MS format, apply the K-to-Jy factor and perform imaging
8	hsd_importdata	Import data into the single dish pipeline
9	hsd_inspectdata	Inspect the whole dataset registered to context
10	hsd_mstoscantable	Convert from MS to Scantable format
11	hsd_plotflagbaseline	Generate plot to summarize latest baseline subtraction and flagging
12	hsd_simplescale	Multiply spectra by a scaling factor

# 5 Common Task Descriptions

## 5.1 *h\_init*

*h\_init* must be called before any other interferometry pipeline task. The pipeline can be initialised in one of two ways: by creating a new pipeline state (*h\_init*) or be loading a saved pipeline state (*h\_resume*). *h\_init* creates an empty pipeline context but does not load visibility data into the context. *hif\_importdata* can be used to load interferometry data.

### Task Description

Initialise the interferometry pipeline  
The *h\_init* task initialises the interferometry pipeline.

### Keyword arguments:

#### ---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode' or 'getinputs' modes

**loglevel** -- Pipeline log level threshold: (debug|info|warning|error|critical).  
Log messages below this threshold will not be displayed.  
default: 'info'

**plotlevel** -- Pipeline plot level threshold: (all|summary).  
Switch between generation of all plots ('all') or just summary plots ('summary'), omitting the per antenna/spw/field plots and detail pages from the web log.  
default: 'all'

**output\_dir** -- Working directory for pipeline processing. Some pipeline processing products such as HTML logs and images will be directed to subdirectories of this path.  
default: '.' (current directory)  
overwrite -- Overwrite existing MSs on input.

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the

pipeline task is returned.

## Examples

1. Create the pipeline context  
`h_init()`

## Parameter List

**Table 5: h\_init default settings**

Parameter	Type	Default	Description
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>loglevel</b>	string	info	Log level for pipeline messages
<b>plotlevel</b>	string	all	Level for pipeline plots
<b>output_dir</b>	string	./	The output working directory
<b>overwrite</b>	bool	True	Overwrite existing files on import
<b>dryrun bool</b>	bool	False	Run the task (False) or display the task command (True)
<b>acceptresults</b>	bool	True	Add the results into the Pipeline context

## 5.2 *h\_resume*

`h_resume` restores a named pipeline state from disk allowing a suspended pipeline reduction session to be resumed.

### Task Description

Restore a save pipeline state from disk

`h_resume` restores a name pipeline state from disk allowing a suspended pipeline reduction session to be resumed.

### Keyword parameters:

**filename** -- Name of the saved pipeline state. Setting filename to 'last' restores the most recently saved pipeline state whose name begins with 'context\*'.  
default: 'last'

example: `filename='context.s3.2012-02-13T10:49:11'`  
`filename='last'`

## Examples

1. Resume the last saved session  
`h_resume()`
2. Resume the named saved session  
`h_resume(filename='context.s3.2012-02-13T10:49:11')`

## Parameter List

**Table 6: h\_resume default settings**

Parameter	Type	Default	Description
<b>filename</b>	string	last	Filename of saved state to be restored

### 5.3 *h\_save*

h\_save saves the current pipeline state to disk under a unique name. If no name is supplied one is generated automatically from a combination of the rootname 'context', the current stage number, and a timestamp.

#### Task Description

Save the pipeline state to disk

h\_save saves the current pipeline state to disk under a unique name.

#### Keyword arguments:

**filename** -- Name of the saved pipeline state. If filename is " then a unique name will be generated computed from the root 'context', the current stage number, and the timestamp.  
default: "

#### Examples

1. Save the current state in the default file  
h\_save()

2. Save the current state to a user named file  
h\_save(filename='savestate\_1')

#### Parameter List

**Table 7: h\_save default settings**

Parameter	Type	Default	Description
<b>filename</b>	string	None	Name for saved state

### 5.4 *h\_weblog*

#### Task Description

Open the pipeline weblog in a browser

#### Parameter List

No parameters

# 6 Interferometry Common Task Descriptions

## 6.1 *hif\_antpos*

The *hif\_antpos* task corrects the antenna positions recorded in the ASDMs using updated antenna position calibration information determined after the observation was taken. Corrections can be input by hand, read from a file on disk, or in future by querying an ALMA database service. The corrections are used to generate a calibration table which is recorded in the pipeline context and applied to the raw visibility data, on the fly to generate other calibration tables, or permanently to generate calibrated visibilities for imaging.

### Task Description

Derive an antenna position calibration table

#### Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context dependent pipeline inputs automatically.

In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.  
default: 'automatic'.

#### ---- pipeline parameters that can be set in any pipelinemode

**hm\_antpos** -- Heuristics method for retrieving the antenna position corrections. The options are 'online' (not yet implemented), 'manual', and 'file'.

default: 'manual'

example: hm\_antpos='file'

**antenna** -- The list of antennas for which the positions are to be corrected if hm\_antpos is 'manual'

default: none

example 'DV05,DV07'

**offsets** -- The list of antenna offsets for each antenna in 'antennas'. Each offset is a set of 3 floating point numbers separated by commas, specified in the ITRF frame.

default: none

example: [0.01, 0.02, 0.03, 0.03, 0.02, 0.01]

**antposfile** -- The file(s) containing the antenna offsets. Used if hm\_antpos is 'file'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- List of input visibility files

default: []

example: ['ngc5921.ms']

**caltable** -- Name of output gain calibration tables

default: []

example: caltable=['ngc5921.gcal']

#### -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Examples

1. Correct the position of antenna 5 for all the visibility files in a single pipeline run.

```
hif_antpos (antenna='DV05', offsets=[0.01, 0.02, 0.03])
```

2. Correct the position of antennas for all the visibility files in a single pipeline run using antenna positions files on disk. These files are assumed to conform to a default naming scheme if 'antposfile' is unspecified by the user.

```
hif_antpos (hm_antpos='file')
```

### Parameter List

**Table 8: hif\_antpos default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>caltable</b>	stringArray	None	List of output caltable(s)
<b>hm_antpos</b>	string	manual	The antenna position determination method
<b>antenna</b>	string	None	List of antennas to be corrected
<b>offsets</b>	doubleArray	None	List of position corrections one set per antenna
<b>antposfile</b>	string	None	File containing antenna position corrections
<b>pipelinemode</b>	string	automatic	The pipeline operation mode
<b>dryrun</b>	bool	False	Run the task (False) or list commands (True)
<b>acceptresults</b>	bool	True	Automatically accept results into context

## 6.2 *hif\_applycal*

*hif\_applycal* applies the precomputed calibration tables stored in the pipeline context to the set of visibility files using predetermined field and spectral window maps and default values for the

interpolation schemes.

Users can interact with the pipeline calibration state using the tasks hif\_export\_calstate and hif\_import\_calstate.

## Task Description

Apply the calibration(s) to the data

Apply precomputed calibrations to the data.

### ---- pipeline parameters that can be set in any pipelinemode

**applymode** -- Calibration apply mode

''='calflagstrict': calibrate data and apply flags from solutions using the strict flagging convention

'trial': report on flags from solutions, dataset entirely unchanged

'flagonly': apply flags from solutions only, data not calibrated

'calonly': calibrate data only, flags from solutions NOT applied

'calflagstrict':

'flagonlystrict': same as above except flag spws for which calibration is unavailable in one or more tables (instead of allowing them to pass uncalibrated and unflagged)

default: ''

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically.

In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.

default: 'automatic'.

### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets in the pipeline context.

default: []

example: ['X227.ms']

**field** -- A string containing the list of field names or field ids to which the calibration will be applied. Defaults to all fields in the pipeline context.

default: ''

example: '3C279', '3C279, M82'

**intent** -- A string containing a the list of intents against which the selected fields will be matched. Defaults to all supported intents in the pipeline context.

default: ''

example: '\*TARGET\*'

**spw** -- The list of spectral windows and channels to which the calibration will be applied. Defaults to all science windows in the pipeline context.

default: ''

example: '17', '11, 15'

**antenna** -- The list of antennas to which the calibration will be applied. Defaults to all antennas. Not currently supported.

### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned

### Examples

1. Apply the calibration to the target data  
hif\_applycal (intent='TARGET')

### Parameter List

Table 9: hif\_applycal default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	none	List of input measurement sets
<b>field</b>	string	none	Set of data selection field names or ids
<b>intent</b>	string	none	Set of data selection observing intents
<b>spw</b>	string	none	Set of data selection spectral window/channels
<b>antenna</b>	string	none	Set of data selection antenna ids
<b>applymode</b>	string	none	Calibration mode: ""="calflagstrict", "calflag", "calflagstrict", "trial", "flagonly", "flagonlystrict", or "calonly"
<b>calwt</b>	boolArray	true	Calibrate the weights as well as the data
<b>flagbackup</b>	bool	true	Backup the flags before the apply
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run task (False) or display the command(True)
<b>acceptresults</b>	bool	True	Automatically accept results into the context

## 6.3 *hif\_atmflag*

Spectral window channels with low atmospheric transmission are identified and flagged. The flagging view comprises a transmission spectrum for each spectral window calculated using the CASA atmosphere model.

Flags are generated by running the following rules on each spectrum:

If flag\_minabs = True then channels with transmission below fmin\_limit are flagged.

If flag\_nmedian = True then channels with transmission below fnm\_limit \* median transmission are flagged.

The flagging limits are set by frequency rather than by channel number.

The frequency frame is the native one of the spectral windows, usually TOPO.

### Task Description

Flag channels with bad atmospheric transmission hif\_atmflag flags channels where the atmospheric transmission is low

### Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically.

In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.

default: 'automatic'.

### ---- pipeline parameters that can be set in any pipelinemode

**intents** -- Specifies the data intents whose channels are to be flagged if they have low atmospheric transmission. This string is inserted into the flagcmd given to the flagdata task applying the flags; it must have a valid flagcmd format.  
default '\*AMP\*,\*BANDPASS\*,\*PHASE\*'

**flag\_minabs** -- True to flag channels with transmission < fmin\_limit.  
default False

**fmin\_limit** -- The atmospheric transmission below which channels are to be flagged if flag\_minabs is True.  
default 0.1

**flag\_nmedian** -- True to flag channels with transmission < fnm\_limit \* median transmission.  
default: False

**fnm\_limit** -- Flag channels with transmission < fnm\_limit \* median transmission, if flag\_nmedian is True.  
default: 0.5

### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- List of input measurement sets  
default: [] - Use the measurement sets currently stored in the pipeline context.  
example: vis=['X132.ms']

### -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: True

**acceptresults** -- This parameter has no effect. The Tsyscal file is already in the pipeline context and is flagged in situ.

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

## Examples

1. Flag channels with transmission below 0.1 in each SpW.

hif\_atmflag()

equivalent to:

hif\_atmflag(flag\_minabs=True, fmin\_limit=0.1)

2. Flag channels with transmission below 0.4 \* median transmission across the spectral window.

hif\_atmflag(flag\_nmedian=True, fnm\_limit=0.4)

## Parameter List

Table 10: hif\_atmflag default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>intent</b>	string	*AMP*, *BANDPASS*, *PHASE*	Data intents to which flags are to be applied
<b>flag_minabs</b>	bool	False	True to flag channels where transmission < fmin_limit
<b>fmin_limit</b>	double	0.1	Transmission limit below which channels are to be flagged
<b>flag_nmedian</b>	bool	False	True to flag channels where transmission < fnm_limit * median transmission
<b>fnm_limit</b>	double	0.5	If flag_nmedian then flag channels where transmission < fnm_limit * median transmission
<b>pipelinemode</b>	string	automatic	The pipeline operations mode
<b>dryrun</b>	bool	False	Run the task (False)

			or list commands(True)
<b>acceptresults</b>	bool	True	Automatically apply results to context

## 6.4 *hif\_bandpass*

*hif\_bandpass* computes a bandpass solution for every specified science spectral window. By default a 'phaseup' pre-calibration is performed and applied on the fly to the data, before the bandpass is computed. The *hif\_refant* task may be used to precompute a prioritized list of reference antennas.

### Task Description

Compute bandpass calibration solutions

Compute amplitude and phase as a function of frequency for each spectral window in each measurement set.

Previous calibration can be applied on the fly.

### Keyword arguments:

#### --- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically.

In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.

default: 'automatic'.

**phaseup** -- Do a phaseup on the data before computing the bandpass solution  
default: True

**phaseupsolint** -- The phase correction solution interval in CASA syntax.

Used when phaseup is True.

default: 'int'

example: 300

**phaseupbw** -- Bandwidth to be used for phaseup. Used when phaseup is True.

default: "

example: " default to entire bandpass, '500MHz' use central 500MHz

**hm\_bandtype** -- The type of bandpass. The options are 'channel' and 'polynomial' for CASA bandpass types = 'B' and 'BPOLY' respectively.

**solint** -- Time and channel solution intervals in CASA syntax.

default: 'inf,7.8125MHz'

example: 'inf,10ch', 'inf'

**maxchannels** -- The bandpass solution smoothing factor in channels. The solution interval is bandwidth / 240. Set to 0 for no smoothing.

default: 240

example: 0

**combine** -- Data axes to combine for solving. Axes are "", 'scan', 'spw', 'field' or any comma-separated combination.

default: 'scan'

example: combine='scan,field'

**minblperant** -- Minimum number of baselines required per antenna for each solve  
Antennas with fewer baselines are excluded from solutions. Used for  
hm\_bandtype='channel' only.

default: 4

**minsnr** -- Solutions below this SNR are rejected. Used for hm\_bandtype=  
'channel' only

default: 3.0

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the pipeline context.

default: "

example: ['M51.ms']

**caltable** -- The list of output calibration tables. Defaults to the standard pipeline naming convention.

default: "

example: ['M51.bcal']

**field** -- The list of field names or field ids for which bandpasses are computed. Defaults to all fields.

default: "

example: '3C279', '3C279, M82'

**intent** -- A string containing a comma delimited list of intents against which the selected fields are matched. Defaults to all data with bandpass intent.

default: "

example: '\*PHASE\*'

**spw** -- The list of spectral windows and channels for which bandpasses are computed. Defaults to all science spectral windows.

default: "

example: '11,13,15,17'

**refant** -- Reference antenna names. Defaults to the value(s) stored in the pipeline context. If undefined in the pipeline context defaults to the CASA reference antenna naming scheme.

default: "

example: refant='DV01', refant='DV06,DV07'

**solnorm** -- Normalise the bandpass solutions

default: False

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

#### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

## Examples

1. Compute a channel bandpass for all visibility files in the pipeline context using the CASA reference antenna determination scheme.

`hif_bandpass()`

2. Same as the above but precompute a prioritized reference antenna list

`hif_refant()`

`hif_bandpass()`

## Parameter List

**Table 11: hif\_bandpass default settings**

Parameter	Type	Default	Description
<code>vis</code>	<code>stringArray</code>	<code>None</code>	List of input measurement sets
<code>citable</code>	<code>stringArray</code>	<code>None</code>	List of output caltables
<code>field</code>	<code>string</code>	<code>None</code>	Set of data selection field names or ids
<code>intent</code>	<code>string</code>	<code>None</code>	Set of data selection intents
<code>spw</code>	<code>string</code>	<code>None</code>	Set of data selection spectral window/channels
<code>antenna</code>	<code>string</code>	<code>None</code>	Set of data selection antenna IDs
<code>phaseup</code>	<code>bool</code>	<code>True</code>	Phaseup before computing the bandpass
<code>phaseupsolint</code>	<code>any</code>	<code>int</code>	Phaseup correction solution interval
<code>phaseupbw</code>	<code>string</code>	<code>None</code>	Bandwidth to use for phaseup
<code>hm_bandtype</code>	<code>string</code>	<code>channel</code>	Bandpass solution type
<code>solint</code>	<code>any</code>	<code>Inf</code>	Solution intervals
<code>maxchannels</code>	<code>int</code>	<code>240</code>	The smoothing factor in channels
<code>combine</code>	<code>string</code>	<code>scan</code>	Data axes which to combine for solve (scan, spw, and/or field)
<code>refant</code>	<code>string</code>	<code>None</code>	Reference antenna names
<code>solnorm</code>	<code>bool</code>	<code>True</code>	Normalise the bandpass solution
<code>minblperant</code>	<code>int</code>	<code>4</code>	Minimum baselines per antenna required for solve
<code>minsnr</code>	<code>double</code>	<code>3.0</code>	Reject solutions below this SNR

<b>degamp</b>	variant	None	Degree for polynomial amplitude solution
<b>dephase</b>	variant	None	Degree for polynomial phase solution
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## 6.5 *hif\_bpflagchans*

Deviant channels in bandpass calibrations are detected by analysis of the bandpass calibration amplitudes.

Flags are generated by running the following rules on each spectrum:

If flag\_diffmad is True then channels bracketing unusually large jumps in the spectrum are detected and flagged.

If flag\_tmf is True then all channels will be flagged if a substantial proportion of channels have already been flagged for other reasons.

If flag\_edges is True then channels at the band edges are detected and flagged.

If flag\_sharps is True then channels covering sharp spectral features are flagged and the flagging is extended in an attempt to cover the lower flanks of the features as well.

### Task Description

Flag deviant channels in bandpass calibration

*hif\_bpflagchans* flags deviant channels in the bandpass calibration table.

### Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically.  
In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.  
default: 'automatic'.

### ---- pipeline parameters that can be set in any pipelinemode

**flag\_diffmad** -- True to flag channels whose difference is greater than diffmad\_limit \* the MAD of the channel to channel differences across the spectrum.  
default: True

**diffmad\_limit** -- If flag\_diffmad is True then channel pairs will be flagged whose difference is greater than diffmad\_limit \* the MAD of the difference spectrum.  
default: 14

**diffmad\_nchan\_limit** -- If flag\_diffmad is True and the number of channels flagged by diffmad\_limit exceeds diffmad\_nchan\_limit then flag all channels.  
default: 10000

**flag\_tmf** -- True to flag all channels if the proportion of channels already flagged is greater than tmf\_frac\_limit or if the number of channels already flagged is greater than tmf\_nchan\_limit.  
default: False

**tmf\_frac\_limit** -- If flag\_tmf is True then all channels will be flagged if proportionally more than tmf\_frac\_limit of them are already flagged.  
default: 0.05

**tmf\_nchan\_limit** -- If flag\_tmf is True then all channels will be flagged if more than tmf\_nchan\_limit of them are already flagged.  
default: 0.05

**flag\_edges** -- True to flag edges of spectra.

default: False

**edge\_limit** -- The first channels inward from each end of the spectrum where the channel to channel difference falls below 'edge\_limit' times the median across the spectrum are designated as the 'edges'. These channels and those outside them are flagged. If the 'edges' lie more than 1/4 of the way across the spectrum then no flagging is done.  
default: 3

**flag\_sharps** -- True to flag channels that cover sharp spectral features. Uses the same algorithm as flag\_sharps to flag the cores of the features but extends the flagging over the feature flanks until the channel to channel difference falls below 2 \* the median over the spectrum.  
default: False

**sharps\_limit** -- Flag as sharp feature cores those channels bracketing a channel to channel difference > sharps\_limit.  
default: 0.05  
---- pipeline-defined parameters which can be set only in 'interactive mode'

**caltable** -- List of input bandpass calibration tables  
default: [] - Use the table currently stored in the pipeline context.

## -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: True

**acceptresults** -- This parameter has no effect. The Tsycal file is already in the pipeline context and is flagged in situ.

## Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

## Examples

1. Flag birdies in the bandpass calibration for each antenna/SpW.  
Flag the entire calibration spectrum if more than 0.05 of channels or  
more than 4 channels in total are flagged.

`hif_bpflagchans()`

equivalent to:

`hif_bpflagchans(flag_diffmad=True, diffmad_limit=7,  
diffmad_nchan_limit=4)`

2. Flag birdies in the bandpass calibration for each antenna/SpW.

`hif_bpflagchans(flag_tmf=False)`

equivalent to:

`hif_bpflagchans(flag_diffmad=True, diffmad_limit=7,  
diffmad_nchan_limit=10000)`

## Parameter List

**Table 12: hif\_bpflagchans default settings**

Parameter	Type	Default	Description
<code>caltable</code>	stringArray	None	List of input caltables
<code>flag_hilo</code>	bool	True	True to flag outlier channels
<code>fhl_limit</code>	double	7	Flag channels further from median than limit * MAD
<code>fhl_minsample</code>	double	5	Minimum number of points in sample
<code>flag_tmf</code>	bool	True	True to flag all channels if proportion of channels flagged > <code>tmf_limit</code>
<code>tmf_limit</code>	double	0.3	Fraction of channels flagged that triggers flagging of all channels
<code>pipelinemode</code>	string	automatic	The pipeline operations mode
<code>dryrun</code>	bool	False	Run the task (False) or list commands (True)
<code>acceptresults</code>	bool	True	Automatically apply results to context

## 6.6 `hif_cleanlist`

### Task Description

Compute clean map

Compute a cleaned image for a particular target source/intent and spectral window.

### Keyword arguments:

--- pipeline parameters that can be set in any `pipelinemode`

`pipelinemode` -- The pipeline operating mode. In 'automatic' mode the pipeline

determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline context defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the h\_init or hif\_importdata sets.

example: vis='ngc5921.ms'

vis=['ngc5921a.ms', 'ngc5921b.ms', 'ngc5921c.ms']

default: use all measurement sets in the context

**weighting** -- Weighting to apply to visibilities:

default='natural'; example: weighting='uniform';

Options: 'natural', 'uniform', 'briggs',

'superuniform', 'briggsabs', 'radial'

**weighting\_robust** -- For weighting='briggs' and 'briggsabs'

default=0.0; example: robust=0.5;

Options: -2.0 to 2.0; -2 (uniform)/+2 (natural)

**weighting\_noise** -- For weighting='briggsabs' noise parameter to use for Briggs "abs" weighting  
example noise='1.0mJy'

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

#### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Parameter List

Table 13: hif\_cleanlist default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>target_list</b>	any	{}	Dictionary specifying targets to be imaged; blank will read list from context
<b>weighting</b>	string	natural	Weighting of uv (natural, uniform, briggs, ...)
<b>robust</b>	double	0.0	Briggs robustness

			parameter
<b>noise</b>	any	1.0Jy	noise parameter for briggs abs mode weighting
<b>npixels</b>	int	1	number of pixels for superuniform or briggs weighting
<b>hm_masking</b>	string	None	Pipeline heuristics masking option
<b>hm_cleaning</b>	string	None	Pipeline cleaning mode
<b>tlimit</b>	double	2.0 T	imes the sensitivity limit for cleaning
<b>masklimit</b>	int	4	Times good mask pixels for cleaning
<b>maxncleans</b>	int	1	Maximum number of clean task calls
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command(True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## 6.7 *hif\_clean*

### Task Description

Compute clean map

Compute a cleaned image for a particular target source/intent and spectral window.

### Keyword arguments:

#### --- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline context defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.

default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets in the context.

default: ''

example: vis=['ngc5921a.ms', ngc5921b.ms', 'ngc5921c.ms']

**imagename** -- Prefix of output images. Defaults to one of the following options depending on the availability of project information.

'{ousstatus uid}.{field}.[{intent}].s{stage number}.spw{spw}'  
'multivis.{field}.[{intent}].s{stage number}.spw{spw}'  
cleanboxes and thresholds to use as it goes. For each iteration the  
output images are:  
{prename}.iter{n}.image; cleaned and restored image  
{prename}.iter{n}.psf; point spread function (dirty beam)  
{prename}.iter{n}.flux; relative sky sensitivity over field  
{prename}.iter{n}.flux.pbcoverage; relative pb coverage over field  
(only for mosaics)  
{prename}.iter{n}.model; image of clean components  
{prename}.iter{n}.residual; image of residuals  
{prename}.iter{n}.cleanmask; image of cleanmask used  
default: ""  
example: 'test1'

**intent** -- An intent against which the selected fields are matched. Default  
means select all data from fields specified by 'field' parameter  
default: ""  
example: "", "TARGET"

**field** -- Fields id(s) or name(s) to image or mosaic. Must be set.  
default:  
example: '3C279', 'Centaurus\*'

**spw** -- Spectral window/channels to image. '\\" for all science data.  
default: ""

example: '9', '9,11'  
mode -- Frequency imaging mode, 'mfs', 'frequency'. '\\" defaults to  
'frequency' if intent parameter includes 'TARGET' otherwise 'mfs'.  
default: ""  
example: 'mfs', 'mosaic'

**imagermode** -- Advanced imaging mode e.g. mosaic or Cotton-Schwab clean.  
Derived as follows:

1. The 'field' parameter is converted into a list of field\_ids for each measurement set in 'vis'.
2. If there is more than 1 field\_id in the list for any measurement set then imagermode is set to  
'mosaic', otherwise it will be set to 'csclean'.

default: ""

**outframe** -- The reference frame of the output image. The only supported option  
is 'LSRK'

default: ""

example: 'LSRK'

**imsiz**e -- X and Y image size in pixels). Must be even and contain factors  
2,3,5,7 only.

Default derived as follows:

1. Determine 'phasecenter' value and spread of field centres around it.
2. Set size of image to cover spread of field centres plus a border of  
width 0.75 \* beam radius (to first null).
3. Divide x and y extents by 'cell' values to arrive at the numbers of  
pixels required.

default: ""

example: [320,320]

**cell** -- X and Y cell size. Derived from maximum UV spacing. Details TBD  
default "

example: ['0.5arcsec', '0.5arcsec']

**phasecenter** -- Direction measure or field id for the mosaic center.

Default derived as follows:

1. Make an array containing all the field centers to be imaged together.
2. Derive the mean direction from the directions array.

default: '\\'

example: 2

**nchan** -- Number of channels or planes in the output image, -1 for all

default: -1

example: 128

**width** -- Width of spectral dimension in frequency, '\\' for default.

default: '\\'

example: '7.8125MHz'

**weighting** -- Weighting to apply to visibilities. Options are: 'natural',

'uniform', 'briggs', 'superuniform', 'briggsabs', 'radial'

default='natural'

example: weighting='uniform'

**robust** -- Parameter for 'briggs' and 'briggsabs' weighting. Ranges from -2.0 to 2.0. -2 for uniform +2 for natural.

default=0.0

example: 0.5

**noise** -- Parameter for 'briggsabs' weighting

default: '1.0Jy'

example: '0.5Jy'

**npixels** -- Parameter for 'briggs' and 'briggsabs'; weighting

default: 1

example: 1

**restoringbeam** -- Gaussian sestoring beam for clean, '\\' for default

default: '\\'

example:

**hm\_masking** -- Clean masking mode. Options are 'none', 'centralquarter'

'psf', 'psfilter' and 'manual'

default: 'centralquarter'

example: 'manual'

**mask** -- Image mask for hm\_masking manual mode. User responsible for matching image sizes, coordinates, etc.

default: '\\'

example: 'mymask.mask'

**niter** -- Maximum number of iterations per clean call

default: 500

example: 100

**threshold** -- Threshold for cleaning

default: '0.0'

example: '0.05'

**maxncleans** -- Maximum number of clean calls

default: 1

example: 10

### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

## Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

## Examples

Make an 'mfs' image of calibrator 3c279 using data in spectral window 1.

The cell size is set to 0.2 arcsec in RA and Dec. Other clean parameters are derived from heuristics:

```
hif_clean(field='3c279', cell='0.2arcsec', spw='1', mode='mfs')
```

Make a cube of calibrator 3c279 using data in spectral window 1. The cube planes will be evenly spaced in frequency in the LSRK frame. Other clean parameters are derived from heuristics.

```
hif_clean(field='3c279', cell='0.2arcsec', spw='1', mode='frequency',  
outframe='LSRK')
```

## Parameter List

Table 14: hif\_clean default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets, '\'' for default
<b>imagename</b>	string	None	Prefix for image filenames, '\'' for default
<b>intent</b>	string	None	Set of data selection intents, '\'' for all
<b>field</b>	string	None	Set of data selection field names or ids
<b>spw</b>	string	None	Set of data selection spectral window/channels '\'' for all
<b>uvrange</b>	any	None	Set of uv ranges, '\'' for all
<b>mode</b>	string	None	Spectral gridding type (mfs, frequency, '\'' for default)
<b>imagermode</b>	string	None	Imaging mode (csclean, mosaic, '\'' for default)
<b>outframe</b>	string	None	velocity frame of output image (LSRK, '\'' for default)
<b>imsize</b>	intArray	None	Y and Y image size in

			pixels, single value same for both, `\\` for default
<b>cell</b>	stringArray	None	Y and Y cell size(s), single value same for both, `\\` for default
<b>phasecenter</b>	any	None	Image center (direction or field index), `\\` for default
<b>nchan</b>	int	-1	Number of channels or planes in output image, -1 = all
<b>start</b>	any	None	Start of output spectral dimension
<b>width</b>	any	None	Width of output spectral channels, `\\` for default
<b>weighting</b>	string	natural	Type of weighting
<b>robust</b>	double	0.0	Briggs weighting robustness parameter
<b>noise</b>	any	1.0Jy	Briggs weighting noise parameter
<b>npixels</b>	int	1	Weighting algorithm parameter
<b>restoringbeam</b>	stringArray	None	Gaussian restoring beam, `\\` for default
<b>hm_masking</b>	string	none	Pipeline heuristics masking option
<b>hm_cleaning</b>	string	manual	Pipeline clean control heuristics
<b>mask</b>	any	None U	ser mask, `\\` for whole image
<b>niter</b>	Int	500	Maximum number of clean iterations
<b>threshold</b>	double	0.0	Flux level to stop cleaning, must include units: `'1.0mJy`'
<b>tlimit</b>	double	2.0	Times the sensitivity limit for cleaning
<b>masklimit</b>	int	4	Times good mask pixels for cleaning
<b>maxncleans</b>	int	1	Maximum number of clean task calls
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## 6.8 *hif\_export\_calstate*

*hif\_export\_calstate* saves the current pipeline calibration state to disk in the form of a set of

equivalent applycal calls. If filename is not given, hif\_export\_calstate saves the calibration state to disk with a filename based on the pipeline context creation time, using the extension '.calstate'. One of two calibration states can be exported: either the active calibration state (those calibrations currently applied on-the-fly but scheduled for permanent application to the measurement set in a subsequent hif\_applycal call) or the applied calibration state (calibrations that were previously applied to the measurement set using hif\_applycal). The default is to export the active calibration state.

## Task Description

Save the pipeline calibration state to disk  
hif\_export\_calstate saves the current pipeline calibration state to disk in the form of a set of equivalent applycal calls.

### Keyword arguments:

**filename** -- Name for the saved calibration state.  
**state** -- calibration state to export

### Examples

1. Save the calibration state.

hif\_export\_calstate()

2. Save the active calibration state with a custom filename

hif\_export\_calstate(filename='afterbandpass.calstate')

3. Save the applied calibration state with a custom filename

hif\_export\_calstate(filename='applied.calstate', state='applied')

### Parameter List

Table 15: hif\_export\_calstate default settings

Parameter	Type	Default	Description
<b>filename</b>	string	None	Name for saved calibration state
<b>state</b>	string	active	The calibration state to export

## 6.9 hif\_exportdata

The hif\_exportdata task exports the data defined in the pipeline context and exports it to the data products directory, converting and or packing it as necessary. The current version of the task exports the following products

- o an XML file containing the pipeline processing request
- o a tar file per ASDM / MS containing the final flags version
- o a text file per ASDM / MS containing the final calibration apply list
- o a FITS image for each selected calibrator source image
- o a FITS image for each selected science target source image
- o a tar file per session containing the caltables for that session
- o a tar file containing the file web log
- o a text file containing the final list of CASA commands
- o a python script that can be used to calibrate the data
- o a python script that can be used to restore raw data directly to calibrated measurement

set

## Task Description

Prepare interferometry data for export

The hif\_exportdata task exports the data defined in the pipeline context and exports it to the data products directory, converting and or packing it as necessary.

### Keyword arguments:

---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In 'interactive' mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- List of visibility data files for which flagging and calibration information will be exported. Defaults to the list maintained in the pipeline context.

default: []

example: vis=['X227.ms', 'X228.ms']

**sessions** -- List of sessions one per visibility file. Currently defaults to a single virtual session containing all the visibility files in vis. In future will default to set of observing sessions defined in the context.

default: []

example: sessions=['session1', 'session2']

**pprfile** -- Name of the pipeline processing request to be exported. Defaults to a file matching the template 'PPR\_\*.xml'.

default: []

example: pprfile=['PPR\_GRB021004.xml']

**calintents** -- List of calibrator image types to be exported. Defaults to all standard calibrator intents 'BANDPASS', 'PHASE', 'FLUX'

default: ''

example: calintents='PHASE'

**calimages** -- List of calibrator images to be exported. Defaults to all calibrator images recorded in the pipeline context.

default: []

example: calimages=['3C454.3.bandpass', '3C279.phase']

**targetimages** -- List of science target images to be exported. Defaults to all science target images recorded in the pipeline context.

default: []

example: targetimages=['NGC3256.band3', 'NGC3256.band6']

**products\_dir** -- Name of the data products subdirectory. Defaults to './'

default: ''

example: products\_dir='..products'

### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Examples

1. Export the pipeline results for a single sessions to the data products directory

!mkdir ..products

hif\_exportdata (products\_dir='..products')

2. Export the pipeline results to the data products directory specify that only the gain calibrator images be saved.

!mkdir ..products

hif\_exportdata (products\_dir='..products', calintents='\*PHASE\*')

### Parameter List

Table 16: hif\_exportdata default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input visibility data
<b>sessions</b>	stringArray	None	List of sessions one per visibility file
<b>pprfile</b>	string	None	The pipeline processing request file to be exported
<b>calintents</b>	string	None	The calibrator source target intents to be exported
<b>calimages</b>	stringArray	None	List of calibrator images to be exported
<b>targetimages</b>	stringArray	None	List of target images to be exported
<b>products_dir</b>	string	None	The data products directory
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display task command (True)

<b>acceptresults</b>	bool	True	Add the results into the pipeline con
----------------------	------	------	---------------------------------------

## 6.10 *hif\_findcont*

### Task Description

Find continuum frequency ranges Compute continuum ranges for all sources and spectral windows.

#### --- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.

default: 'automatic'. ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the h\_init or hif\_importdata sets.

example: vis='ngc5921.ms'

vis=['ngc5921a.ms', ngc5921b.ms', 'ngc5921c.ms'] default: use all measurement sets in the context

**parallel** -- use multiple CPU nodes to compute dirty images default: 'automatic'

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False acceptresults -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True Output: results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned. Examples:

### Parameter List

Table 17: *hif\_findcont* default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>target_list</b>	any	{}	Dictionary specifying targets to be imaged; blank will read list from context
<b>parallel</b>	string	automatic	Compute dirty images using MPI cluster
<b>pipelinemode</b>	string	automatic	The pipeline operating mode

<b>dryrun</b>	bool	False	Run the task (False) or display the command(True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## 6.11 *hif\_gaincal*

The complex gains are derived from the data column (raw data) divided by the model column (usually set with hif\_setjy). The gains are obtained for a specified solution interval, spw combination and field combination. Good candidate reference antennas can be determined using the hif\_refant task. Previous calibrations that have been stored in the pipeline context are applied on the fly. Users can interact with these calibrations via the hif\_export\_calstate and hif\_import\_calstate tasks.

### Task Description

Determine temporal gains from calibrator observations  
Compute the gain solutions.

#### ---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline context defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

**hm\_gtype** -- The type of gain calibration. The options are 'gtype' and 'gspline' for CASA gain types = 'G' and 'GSPLINE' respectively.  
**calmode** -- Type of solution, The options are 'ap' (amp and phase), 'p' (phase only) and 'a' (amp only)  
default: 'ap'  
options: 'p','a','ap'

**solint** -- Time solution intervals in CASA syntax. Works for hm\_gtype='gtype' only.  
default: 'inf'  
example: 'inf', 'int', '100sec'  
**combine** -- Data axes to combine for solving. Options are "", 'scan', 'spw', 'field' or any comma-separated combination. Works for hm\_gtype='gtype' only.  
default: ""  
example: combine=""

**minblperant** -- Minimum number of baselines required per antenna for each solve  
Antennas with fewer baselines are excluded from solutions. Works for hm\_gtype='gtype' only.  
default: 4  
example: minblperant=2  
**minsnr** -- Solutions below this SNR are rejected. Works for hm\_gtype='channel' only  
default: 3.0

**splinetime** -- Spline timescale (sec). Used for hm\_gtype='gspline'. Typical

splinetime should cover about 3 to 5 calibrator scans.

default: 3600 (1 hour)

example: splinetime=1000

npointaver -- Tune phase-unwrapping algorithm. Used for hm\_gtype='gspline'

default: 3 (Keep at this value)

phasewrap -- Wrap the phase for changes larger than this amount (degrees)

Used for hm\_gtype='gspline'.

default: 180 (Keep at this value)

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the pipeline context

default: "

example: ['M82A.ms', 'M82B.ms']

**caltable** -- The list of output calibration tables. Defaults to the standard pipeline naming convention.

default: "

example: ['M82.gcal', 'M82B.gcal']

**field** -- The list of field names or field ids for which gain solutions are to be computed. Defaults to all fields with the standard intent.

default: "

example: '3C279', '3C279, M82'

**intent** -- A string containing a comma delimited list of intents against which the selected fields are matched. Defaults to \*PHASE\*.

default: "

example: "", "\*AMP\*,\*PHASE\*"

**spw** -- The list of spectral windows and channels for which gain solutions are computed. Defaults to all science spectral windows.

default: "

example: '3C279', '3C279, M82'

**smodel** -- Point source Stokes parameters for source model (experimental)

Defaults to using standard MODEL\_DATA column data.

default: []

example: [1,0,0,0] (I=1, unpolarized)

**refant** -- Reference antenna name(s) in priority order. Defaults to most recent values set in the pipeline context. If no reference antenna is defined in the pipeline context use the CASA defaults.

default: "

example: refant='DV01', refant='DV05,DV07'

**solnorm** -- Normalise the gain solutions

default: False

--- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

#### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise

the results object for the pipeline task is returned

## Examples

1. Compute standard per scan gain solutions that will be used to calibrate the target.

`hif_gaincal()`

## Parameter List

**Table 18: hif\_gaincal default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>caltable</b>	stringArray	None	List of output caltables
<b>field</b>	string	None	Set of data selection field names or ids
<b>intent</b>	string	None	Set of data selection observing intents
<b>spw</b>	string	None	Set of data selection spectral window/channels
<b>antenna</b>	string	None	Set of data selection antenna ids
<b>hm_gaintype</b>	string	gtype	The gain solution type (gtype or gpsline)
<b>calmode</b>	string	ap	Type of solution" (ap, p, a)
<b>solint</b>	any	inf	Solution intervals
<b>combine</b>	string	None	Data axes which to combine for solve (scan, spw, and/or field)
<b>refant</b>	string	None	Reference antenna names
<b>solnorm</b>	bool	False	Normalize average solution amplitudes to 1.0
<b>minblperant</b>	int	4	Minimum baselines per antenna required for solve
<b>minsnr</b>	double	3.0 R	eject solutions below this SNR
<b>smodel</b>	doubleArray	None	Point source Stokes parameters for source model
<b>splinetime</b>	double	3600.0	Spline timescale(sec)
<b>npointaver</b>	int	3	The phase-unwrapping algorithm
<b>phasewrap</b>	double	180.0	Wrap the phase for jumps greater than

			this value (degrees)
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Automatically accept results into the context

## 6.12 hif\_gainflag

### Task Description

Flag antennas with deviant gain. hif\_gainflag flags data for antennas with deviant median gains and/or high gain rms. Keyword arguments: pipelinemode -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task. default: 'automatic'.

#### ---- pipeline parameters that can be set in any pipelinemode

**intent** -- The data intent to be checked for antennas with deviant gains. The default is blank, which causes the task to select the first intent in the list ['BANDPASS', 'PHASE', 'AMPLITUDE'] for which data are found.

default "

**flag\_mediandeviant** -- True to flag antennas with deviant median gain, calculated as:

$\text{abs}(\text{median}(\text{antenna}) - \text{median}(\text{all antennas})) / \text{MAD}(\text{all antennas}) > \text{fmeddev\_limit}$  default False

**fmeddev\_limit** -- Flag antennas with '\mediandeviant' metric larger than fmeddev\_limit default 3.0

**flag\_rmsdeviant** -- True to flag antennas with deviant gain rms, calculated as:  $\text{stdev}(\text{antenna}) / \text{MAD}(\text{all antennas}) > \text{frmsdev\_limit}$

default True

**frmsdev\_limit** -- Flag antennas with '\rmsdeviant' metric larger than frmsdev\_limit

default 3.5

**metric\_order** -- Order in which the flagging metrics are evaluated. default 'mediandeviant, rmsdeviant'

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

#### -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True **acceptresults** -- Add the results of the task to the pipeline context (True) or reject

them (False).

default: True

#### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned. Description Deviant antennas are detected by analysis of a view showing their calibration gains. Two flagging metrics are available (mediandeviant, rmsdeviant) and a flagging view is created for each metric that has been enabled. Each view is a list of 2D images with axes 'Time' and 'Antenna'; there is one image for each spectral window and intent. If either of the flagging metrics exceeds their corresponding limits (fmeddev\_limit, frmsdev\_limit) for a given antenna in a given spw, then all data for that antenna and for all spws within the same baseband will be flagged with a flagcmd. The following heuristics are used for the two flagging metrics:

mediandeviant:

$\text{abs}(\text{median}(\text{antenna}) - \text{median}(\text{all antennas})) / \text{MAD}(\text{all antennas}) > \text{fmeddev\_limit}$  rmsdeviant:

$\text{stdev}(\text{antenna}) / \text{MAD}(\text{all antennas}) > \text{frmsdev\_limit}$  where MAD is the median absolute deviation from the median.

#### Parameter List

Table 18: hif\_gainflag default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>intent</b>	string	None	Data intent whose gains are to checked
<b>spw</b>	string	None	Spectral window ids whose gains are to be checked
<b>refant</b>	string	None	Reference antenna names
<b>flag_mediandeviant</b>	bool	False	True to flag antennas with deviant median gains
<b>fmeddev_limit</b>	double	3.0	Flag antennas with '\mediandeviant\' metric above fmeddev_limit
<b>flag_rmsdeviant</b>	bool	True	True to flag antennas with deviant gain rms
<b>frmsdev_limit</b>	double	3.5	Flag antennas with '\rmsdeviant\' metric above frmsdev_limit
<b>metric_order</b>	string	mediandeviant, rmsdeviant	Order in which the flagging metrics are evaluated
<b>pipelinemode</b>	string	automatic	The pipeline operations mode
<b>dryrun</b>	bool	False	Run the task (False) or list

			commands(True)
<b>acceptresults</b>	bool	True	Automatically apply results to context

## 6.13 *hif\_import\_calstate*

*hif\_import\_calstate* clears and then recreates the pipeline calibration state based on the set of *applycal* calls given in the named file. The *applycal* statements are interpreted in additive fashion; for identically specified data selection targets, *caltables* specified in later statements will be added to the state created by earlier calls.

### Task Description

Import a calibration state from disk  
Import a calibration state to disk.

### Keyword arguments:

**filename** -- Name of the saved calibration state.

### Examples

1. Import a calibration state from disk.  
`hif_import_calstate(filename='aftergaincal.calstate')`

### Parameter List

**Table 19: hif\_import\_calstate default settings**

Parameter	Type	Default	Description
<b>filename</b>	string	None	Name of the saved calibration state

## 6.14 *hif\_importdata*

### Task Description

Imports data into the interferometry pipeline  
The *hif\_importdata* task loads the specified visibility data into the pipeline context unpacking and / or converting it as necessary.

### Keyword arguments:

#### ---- pipeline parameters that can be set in any pipelinemode

**vis** -- List of visibility data files. These may be ASDMs, tar files of ASDMs, MSs, or tar files of MSs. If ASDM files are specified, they will be converted to MS format.

default: []

example: vis=['X227.ms', 'asdms.tar.gz']

**session** -- List of sessions to which the visibility files belong. Defaults to a single session containing all the visibility files, otherwise a session must be assigned to each vis file.

default: []

example: session=['Session\_1', 'Sessions\_2']

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In 'interactive' mode the user can set the pipeline context defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.

default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**asis** -- ASDM tables to convert as is

default: 'Antenna Station Receiver CalAtmosphere'

example: 'Receiver', "

**process\_caldevice** -- Ingest the ASDM caldevice table

default: False

example: True

**overwrite** -- Overwrite existing MSs on output.

default: False

**bdfflags** -- Apply BDF flags on import

default: True

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

#### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

#### Examples

1. Load an ASDM list in the ..../rawdata subdirectory into the context.

```
hif_importdata (vis=['..../rawdata/uid__A002_X30a93d_X43e',
'..../rawdata/uid_A002_x30a93d_X44e'])
```

2. Load an MS in the current directory into the context.

```
hif_importdata (vis=[uid__A002_X30a93d_X43e.ms])
```

3. Load a tarred ASDM in ..../rawdata into the context.

```
hif_importdata (vis=['..../rawdata/uid__A002_X30a93d_X43e.tar.gz'])
```

4. Check the hif\_importdata inputs, then import the data

```

myvislist = ['uid__A002_X30a93d_X43e.ms', 'uid_A002_x30a93d_X44e.ms']
hif_importdata(vis=myvislist, pipelinemode='getinputs')
hif_importdata(vis=myvislist)

```

5. Load an ASDM but check the results before accepting them into the context.

```

results = hif_importdata (vis=['uid__A002_X30a93d_X43e.ms'],
acceptresults=False)
results.accept()

```

6. Run in dryrun mode before running for real

```

results = hif_importdata (vis=['uid__A002_X30a93d_X43e.ms'], dryrun=True)
results = hif_importdata (vis=['uid__A002_X30a93d_X43e.ms'])

```

## Parameter List

**Table 20: hif\_importdata default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input visibility data
<b>session</b>	stringArray	None	List of visibility data sessions
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>asis</b>	string	Antenna Station Receiver CalAtmosphere	ASDM to convert as is
<b>process_caldevice</b>	bool	False	Import the caldevice table from the ASDM
<b>overwrite</b>	bool	False	Overwrite existing files on import
<b>bdfflags</b>	bool	True	Apply BDF flags on import
<b>dryrun</b>	bool	False	Run the task (False) or display task command (True)
<b>acceptresults</b>	bool	True	Add the results into the pipeline Context

## 6.15 hif\_lowgainflag

### Task Description

Flag antennas with low or high gain hif\_lowgainflag flags data for antennas with unusually low or high gains.

### Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.

default: 'automatic'.

#### ---- pipeline parameters that can be set in any pipelinemode

**intent** -- The data intent to be checked for antennas with low gain. The default is blank, which causes the task to select the first intent in the list ['BANDPASS', 'PHASE', 'AMPLITUDE'] for which data are found.

default "

**flag\_nmedian** -- True to flag figures of merit greater than fnm\_hi\_limit \* median or lower than fnm\_lo\_limit \* median.

default True

fnm\_lo\_limit -- Points lower than fnm\_lo\_limit \* median are flagged.  
default 0.7

fnm\_hi\_limit -- Points greater than fnm\_hi\_limit \* median are flagged.  
default 1.3

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

##### -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

##### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

##### Description

Deviant antennas are detected by analysis of a view showing their calibration gains. This view is a list of 2D images with axes 'Time' and 'Antenna'; there is one image for each spectral window and intent.

A flagcmd to flag all data for an antenna will be generated by any gain that is outside the range [fnm\_lo\_limit \* median, fnm\_hi\_limit \* median].

### Parameter List

Table 21: hif\_lowgainflag default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>intent</b>	string	None	Data intent whose gains are to be checked
<b>spw</b>	string	None	Spectral window ids whose gains are to be checked
<b>refant</b>	string	None	Reference antenna names
<b>flag_nmedian</b>	bool	True	True to flag values outside range [fnm_lo_limit * median, fnm_hi_limit * median]
<b>fnm_lo_limit</b>	double	0.7	Flag values lower than

			<code>fnm_lo_limit * median</code>
<b><code>fnm_hi_limit</code></b>	double	1.3	Flag values higher than <code>fnm_hi_limit * median</code>
<b><code>pipelinemode</code></b>	string	automatic	The pipeline operations mode
<b><code>dryrun</code></b>	bool	False	Run the task (False) or list commands (True)
<b><code>acceptresults</code></b>	bool	True	Automatically apply results to context

## ***6.16 hif\_makecleanlist***

Generate a list of images to be cleaned. By default the list will include one image per science target per spw. Calibrator targets can be selected by setting appropriate values for `itent`. By default the output image cellsize is set to the minimum cell size consistent with the UV coverage. By default the image size in pixels is set to values determined by the cell size and the single dish beam size. If a calibrator is being imaged (intents 'PHASE', 'BANDPASS', 'FLUX' or 'AMPLITUDE') then the image dimensions are limited to '`calmaxpix`' pixels.

By default science target images are cubes and calibrator target images are single channel. Science target images may be mosaics or single fields.

### **Task Description**

Compute list of clean images to be produced  
Create a list of images to be cleaned.

### **Keyword Arguments**

**`pipelinemode`** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In 'interactive' mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

#### **--- pipeline parameters that can be set in any `pipelinemode`**

**`mode`** -- Frequency specification:

default: 'mfs'

example:

`mode='mfs'` produce one image from all specified data  
`mode='frequency'`, channels are specified in frequency  
`mode='velocity'`, channels are specified in velocity

**`cell`** -- Cell size (x, y)

default " Compute cell size based on the UV coverage of all the fields to be imaged.

example: ['0.5arcsec', '0.5arcsec']

**`imsize`** -- Image X and Y size in pixels. The sizes must be even and divisible by 2,3,5,7 only.

default: " The default values are derived as follows:

1. Determine phase center and spread of field centers around it.
2. Set the size of the image to cover the spread of field centers plus a border of width  $0.75 * \text{beam radius}$ , to first null.

3. Divide X and Y extents by cell size to arrive at the number of pixels required.

example: [120, 120]

**calmaxpix** -- Maximum image X or Y size in pixels if a calibrator is being imaged ('PHASE', 'BANDPASS', 'AMPLITUDE' or 'FLUX').

default: 300

example: 300

width -- Output channel width.

default: " Difference in frequency between first 2 selected channels.

for frequency mode images.

example: '24.2kHz'

**---- pipeline-defined parameters which can be set only in 'interactive mode'**

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the h\_init or hif\_importdata sets.

default "": use all measurement sets in the context

example: 'ngc5921.ms', ['ngc5921a.ms', ngc5921b.ms', 'ngc5921c.ms']

intent -- Select intents for which associated fields will be imaged.

default: 'TARGET'

example: 'PHASE,BANDPASS'

**field** -- Select fields to image. Use field name(s) NOT id(s). Mosaics are assumed to have common source / field names. If intent is specified only fields with data matching the intent will be selected. The fields will be selected from measurement sets in 'vis'.

default: " Fields matching matching intent, one image per target source.

example: '3C279', 'Centaurus\*', '3C279,J1427-421'

**spw** -- Select spectral window/channels to image.

default: " Individual images will be computed for all science spectral windows.

example: '9'

**uvrange** -- Select a set of uv ranges ro image.

default: " All uv data is included

example: '0~1000klambda', ['0~100klambda', 100~1000klambda]

**phasescenter** -- Direction measure or field id of the image center.

default: " The default phase center is set to the mean of the field directions of all fields that are to be image together.

example: 0, 'J2000 19h30m00 -40d00m00'

**nchan** -- Total number of channels in the output image(s)

default: -1 Selects enough channels to cover the data selected by spw consistent with start and width.

example: 100

**start** -- First channel for frequency mode images.

default " Starts at first input channel of the spw.

example: '22.3GHz'

**--- pipeline task execution modes**

**dryrun** -- Run the command (False) or generate the commands to be run but

do not execute (True).

default: False

`acceptresults` -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

## Output

`results` -- If `pipelinemode` is 'getinputs' then `None` is returned. Otherwise the `results` object for the pipeline task is returned.

## Examples

1. Make a list of science target images to be cleaned, one image per science spw.

`hif_makecleanlist()`

2. Make a list of PHASE and BANDPASS calibrator targets to be imaged, one image per science spw.

`hif_makecleanlist(intent='PHASE,BANDPASS')`

3. Make a list of PHASE calibrator images observed in spw 1, images limited to 50 pixels on a side.

`hif_makecleanlist(intent='PHASE',spw='1',calmaxpix=50)`

## Parameter List

**Table 22: hif\_makecleanlist default settings**

Parameter	Type	Default	Description
<code>vis</code>	stringArray	None	List of input measurement sets
<code>imagename</code>	string	None	Prefix for output image names, '\'' for default.
<code>intent</code>	string	TARGET	Set of data selection intents
<code>field</code>	string	None	Set of data selection field names or ids, '\'' for all
<code>spw</code>	string	None	Set of data selection spectral window/channels, '\'' for all
<code>uvrange</code>	string	None	Set of data selection uv ranges, '\'' for all.
<code>mode</code>	string	None	Spectral gridding type (mfs, frequency, '\'' for default)
<code>outframe</code>	string	None	velocity frame of output image (LSRK, '\'' for default)
<code>imsize</code>	intArray	None	Image X and Y size(s) in pixels, '\'' for default. Single value

			same for both.
<b>cell</b>	stringArray	None	Image X and Y cell size(s), '\\" for default. Single value same for both
<b>calmaxpix</b>	int	300	Maximum X and Y size of calibrator images in pixels
<b>phasecenter</b>	any	None	Image center (direction or field index, '\\" for default)
<b>nchan</b>	int	-1	Number of channels, -1 = all
<b>start</b>	any	None	Channel start, '\\" for default
<b>width</b>	any	None	Channel width, '\\" for default.
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## 6.17 hif\_makeimages

### Task Description

Compute clean map Compute a cleaned image for a particular target source/intent and spectral window.

### Keyword arguments:

#### --- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline context defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task. default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement

sets specified in the h\_init or hif\_importdata sets. example: vis='ngc5921.ms'

vis=['ngc5921a.ms', ngc5921b.ms', 'ngc5921c.ms'] default: use all measurement sets in the context

**weighting** -- Weighting to apply to visibilities: default='briggs'; example: weighting='uniform'; Options: 'natural', 'uniform', 'briggs', 'superuniform', 'briggsabs', 'radial' weighting\_robust -- For weighting='briggs' and 'briggsabs'

default=-999.0; example: robust=0.7; Options: -2.0 to 2.0; -2 (uniform)/+2 (natural)

**weighting\_noise** -- For weighting='briggsabs' noise parameter to use for Briggs "abs" weighting

example noise='1.0mJy' subcontms -- Subtract continuum in MS

Caveat: if enabled, repeated continuum subtraction will lead to erroneous negative fluxes.

default=False

Options: False, True parallel -- use multiple CPU nodes to clean images

default: \\'automatic\'

### **--- pipeline task execution modes**

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False acceptresults -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True Output: results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned. Examples:

## **Parameter List**

**Table 22: hif\_makeimages default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>target_list</b>	any	{}	Dictionary specifying targets to be imaged; blank will read list from context
<b>weighting</b>	string	briggs	Weighting of uv (natural, uniform, briggs, ...)
<b>robust</b>	double	-999.0	Briggs robustness parameter
<b>noise</b>	any	1.0Jy	noise parameter for briggs abs mode weighting
<b>npixels</b>	int	0	number of pixels for superuniform or briggs weighting
<b>hm_masking</b>	string	None	Pipeline heuristics masking option
<b>hm_cleaning</b>	string	None	Pipeline cleaning mode
<b>tlimit</b>	double	2.0	Times the sensitivity limit for cleaning
<b>masklimit</b>	int	4	Times good mask pixels for cleaning
<b>maxnleans</b>	int	1	Maximum number of clean task calls
<b>subcontms</b>	bool	False	Subtract continuum in MS

<b>parallel</b>	string	automatic	Clean images using MPI cluster
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command(True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## **6.18 hif\_makeimlist**

Generate a list of images to be cleaned. By default the list will include one image per science target per spw. Calibrator targets can be selected by setting appropriate values for intent. By default the output image cellsize is set to the minimum cell size consistent with the UV coverage. By default the image size in pixels is set to values determined by the cell size and the single dish beam size. If a calibrator is being imaged (intents 'PHASE', 'BANDPASS', 'FLUX' or 'AMPLITUDE') then the image dimensions are limited to 'calmaxpix' pixels. By default science target images are cubes and calibrator target images are single channel. Science target images may be mosaics or single fields. Issues TBD Examples 1. Make a list of science target images to be cleaned, one image per science spw.

### **Task Description**

Compute list of clean images to be produced Create a list of images to be cleaned.

### **Keyword Arguments**

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline

determines the values of all context defined pipeline inputs automatically. In 'interactive' mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.

default: 'automatic'

### **--- pipeline parameters that can be set in any pipelinemode**

**specmode** -- Frequency specification:

default: 'mfs' example:

mode='mfs' produce one image from all specified data

mode='cube', channels are specified in frequency cell

### **-- Cell size (x, y)**

default " Compute cell size based on the UV coverage of all the fields to be imaged. example: ['0.5arcsec', '0.5arcsec']

**imsize** -- Image X and Y size in pixels. The sizes must be even and divisible by 2,3,5,7 only. default: " The default values are derived as follows:

1. Determine phase center and spread of field centers around it. 2. Set the size of the image to cover the spread of field centers plus

a border of width  $0.75 * \text{beam radius}$ , to first null. 3. Divide X and Y extents by cell size to arrive at the number of pixels required. example: [120, 120]

**calmaxpix** -- Maximum image X or Y size in pixels if a calibrator is being imaged ('PHASE', 'BANDPASS', 'AMPLITUDE' or 'FLUX'). default: 300 example: 300

**width** -- Output channel width. default: " Difference in frequency between first 2 selected channels.

for frequency mode images. example: '24.2kHz'

'pilotimage' for 15 MHz / 8 channel heuristic

**---- pipeline-defined parameters which can be set only in 'interactive mode'**

**vis** -- The list of input measurement sets. Defaults to the list of measurement

sets specified in the h\_init or hif\_importdata sets. default "": use all measurement sets in the context example: 'ngc5921.ms', ['ngc5921a.ms', ngc5921b.ms', 'ngc5921c.ms']

**intent** -- Select intents for which associated fields will be imaged. default: 'TARGET' example: 'PHASE,BANDPASS'

**field** -- Select fields to image. Use field name(s) NOT id(s). Mosaics are assumed to have common source / field names. If intent is specified only fields with data matching the intent will be selected. The fields will be selected from measurement sets in 'vis'. default: " Fields matching matching intent, one image per target source. example: '3C279', 'Centaurus\*', '3C279,J1427-421'

**spw** -- Select spectral window/channels to image. default: " Individual images will be computed for all science spectral windows. example: '9'

**contfile** -- Name of file with frequency ranges to exclude for continuum images. default: 'cont.dat' example: 'mycont.dat'

**linesfile** -- Name of file with line frequency ranges to exclude for continuum images. default: 'lines.dat' example: 'mylines.dat'

**uvrange** -- Select a set of uv ranges ro image. default: " All uv data is included example: '0~1000klambda', ['0~100klambda', 100~1000klambda]

**phasecenter** -- Direction measure or field id of the image center. default: " The default phase center is set to the mean of the field directions of all fields that are to be image together. example: 0, 'J2000 19h30m00 -40d00m00'

**nchan** -- Total number of channels in the output image(s) default: -1 Selects enough channels to cover the data selected by spw consistent with start and width. example: 100

**start** -- First channel for frequency mode images. default " Starts at first input channel of the spw. example: '22.3GHz'

### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False **acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

### Output

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

**hif\_makeimlist()** 2. Make a list of PHASE and BANDPASS calibrator targets to be imaged, one image per science spw.

**hif\_makeimlist(intent='PHASE,BANDPASS')** 3. Make a list of PHASE calibrator images observed in spw 1, images limited to 50 pixels on a side.

**hif\_makeimlist(intent='PHASE',spw='1',calmaxpix=50)**

### Parameter List

**Table 23: hif\_makeimlist default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>imagename</b>	string	None	Prefix for output image names, '\'' for default.
<b>intent</b>	string	TARGET	Set of data selection intents
<b>field</b>	string	None	Set of data selection field names or ids, '\'' for all
<b>spw</b>	string	None	Set of data selection spectral window/channels, '\'' for all
<b>contfile</b>	string	None	Name of file with frequency ranges to use for continuum images.
<b>linesfile</b>	string	None	Name of file with line frequency ranges to exclude for continuum images.
<b>uvrange</b>	string	None	Set of data selection uv ranges, '\'' for all.
<b>specmode</b>	string	None	Spectral gridding type (mfs, cube, '\'' for default)
<b>outframe</b>	string	None	velocity frame of output image (LSRK, '\'' for default)
<b>imsize</b>	intArray	None	Image X and Y size(s) in pixels, '\'' for default. Single value same for both.

<b>cell</b>	stringArray	None	Image X and Y cell size(s), '\\" for default. Single value same for both
<b>calmaxpix</b>	int	300	Maximum X and Y size of calibrator images in pixels
<b>phasescenter</b>	any	None	Image center (direction or field index, '\\" for default)
<b>nchan</b>	int	-1	Number of channels, -1 = all
<b>start</b>	any	None	Channel start, '\\" for default
<b>width</b>	any	None	Channel width, '\\" for default.
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command(True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## 6.19 *hif\_normflux*

Derive flux densities for point source transfer calibrators using flux models for reference calibrators.

Flux values are determined by:

- o computing complex gain phase only solutions for all the science spectral windows using the calibrator data selected by the 'reference' and 'refintent' parameters and the 'transfer' and 'transintent' parameters, and the value of the 'phaseupsolint' parameter.
- o computing complex amplitude only solutions for all the science spectral windows using calibrator data selected with 'reference' and 'refintent' parameters and the 'transfer' and 'transintent' parameters, the value of the 'solint' parameter.
- o transferring the flux scale from the reference calibrators to the transfer calibrators using refspwmap for windows without data in the reference calibrators
- o extracted the computed flux values from the CASA logs and inserting them into the MODEL\_DATA column.

Note that the flux corrected calibration table computed internally is not used in later pipeline apply calibration steps.

### Task Description

Average calibrator fluxes across measurement sets

Derive flux densities for point source transfer calibrators using flux models for reference calibrators.

### ---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the users can check the settings

of all parameters in the task without running the task.  
default: 'automatic'.

**phaseupsolint** -- Time solution intervals in CASA syntax for the phase solution.

default: 'inf'

example: 'inf', 'int', '100sec'

**solint** -- Time solution intervals in CASA syntax for the amplitude solution.

default: 'inf'

example: 'inf', 'int', '100sec'

**---- pipeline-defined parameters which can be set only in  
'interactive mode'**

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the pipeline context

default: ''

example: ['M32A.ms', 'M32B.ms']

**reference** -- A string containing a comma delimited list of field names defining the reference calibrators. Defaults to field names with intent '\*AMP\*'.  
default: ''

example: 'M82,3C273'

**transfer** -- A string containing a comma delimited list of field names defining the transfer calibrators. Defaults to field names with intent '\*PHASE\*'.  
default: ''

example: 'J1328+041,J1206+30'

**refintent** -- A string containing a comma delimited list of intents used to select the reference calibrators. Defaults to \*AMP\*.  
default: ''  
example: "", "\*AMP"

**refspwmap** -- Vector of spectral window ids enabling scaling across spectral windows. Defaults to no scaling  
default: [-1]  
example: [1,1,3,3] (4 spws, reference fields in 1 and 3, transfer fields in 0,1,2,3

**transintent** -- A string containing a comma delimited list of intents defining the transfer calibrators. Defaults to \*PHASE\*.  
default: ''  
example: "", "\*\*PHASE\*\*"

**--- pipeline task execution modes**

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

**Output:**

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned

## Examples

1. Compute flux flux values for the phase calibrator using model data from the amplitude calibrator.

hif\_gfluxscale ()

## Parameter List

Table 24: hif\_gfluxscale default settings

Parameter	Type	Default	Description
<b>vis</b>	string	None	List of input measurements sets
<b>refintent</b>	string	None	Observing intent of reference fields
<b>transintent</b>	string	None	Observing intent of transfer fields
<b>reference</b>	variant	None	Reference calibrator field name(s)
<b>transfer</b>	variant	None	Transfer calibrator field name(s)
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display commands (True)
<b>acceptresults</b>	bool	True	Automatically accept results into context

## 6.20 hif\_rawflagchans

The hif\_rawflagchans task flags deviant baseline/channels in the raw data. The flagging views used are derived from the raw data for the specified intent - default the first of BANDPASS, PHASE and AMPLITUDE to have data. Bad baseline/channels are flagged for all intents, not just the one that is the basis of the flagging views. For each spectral window the flagging view is a 2d image with axes 'channel' and 'baseline'. The pixel for each channel,baseline is the time average of the underlying unflagged raw data. The baseline axis is labelled by numbers of form id1.id2 where id1 and id2 are the IDs of the baseline antennas. Both id1 and id2 run over all antenna IDs in the observation. This means that each baseline is shown twice but has the benefit that 'bad' antennas are easily identified by eye.

Three flagging methods are available:

If parameter flag\_hilo is set True then outliers from the median of each flagging view will be flagged. If parameter flag\_bad\_quadrant is set True then a simple 2 part test is used to check for bad antenna quadrants and/or bad baseline quadrants. Here a 'quadrant' is defined simply as one quarter of the channel axis. The first part of the test is to note as 'suspect' those points further from the view median than fbq\_hilo\_limit \* MAD. The second part is to flag entire antenna/quadrants if their fraction of suspect points exceeds fbq\_antenna\_frac\_limit. Failing that, entire baseline/quadrants may be flagged if their fraction of suspect points exceeds fbq\_baseline\_frac\_limit. Suspect points are not flagged unless as part of a bad antenna or baseline quadrant.

## Task Description

Flag deviant baseline/channels in raw data

### Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.  
default: 'automatic'.

#### ---- pipeline parameters that can be set in any pipelinemode

**flag\_hilo** -- True to flag channel/baseline data further from the view median than fhl\_limit \* MAD.  
default: True

**fhl\_limit** -- If flag\_hilo is True then flag channel/baseline data further from the view median than fhl\_limit \* MAD.  
default: 20

**fhl\_minsample** -- Do no flagging if the view median and MAD are derived from fewer than fhl\_minsample view pixels.  
default: 5

**flag\_bad\_quadrant** -- True to search for and flag bad antenna quadrants and baseline quadrants. Here a '/quadrant' is one quarter of the channel axis.  
default: True

**fbq\_hilo\_limit** -- If flag\_bad\_quadrant is True then channel/baselines further from the the view median than fbq\_hilo\_limit \* MAD will be noted as 'suspect'. If there are enough of them to indicate that an antenna or baseline quadrant is bad then all channel/baselines in that quadrant will be flagged.  
default: 8.0

**fbq\_antenna\_frac\_limit** -- If flag\_bad\_quarant is True and the fraction of suspect channel/baselines in a particular antenna/quadrant exceeds fbq\_antenna\_frac\_limit then all data for that antenna/quadrant will be flagged.  
default: 0.2

**fbq\_baseline\_frac\_limit** -- If flag\_bad\_quarant is True and the fraction of suspect channel/baselines in a particular baseline/quadrant exceeds fbq\_baseline\_frac\_limit then all data for that baseline/quadrant will be flagged.  
default: 1.0 (i.e. no flagging)

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- List of input measurement sets.  
default: [] - Use the measurement sets currently known to the pipeline context.

#### -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- This parameter has no effect. The Tsyscal file is already in the pipeline context and is flagged in situ.

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Examples:

1. Flag bad quadrants and wild outliers, default method.

`hif_rawflagchans()`

equivalent to:

```
hif_rawflagchans(flag_hilo=True, fhl_limit=20,  
flag_bad_quadrant=True, fbq_hilo_limit=8,  
fbq_antenna_frac_limit=0.2, fbq_baseline_frac_limit=1.0)
```

### Parameter List

Table 25: `hif_rawflagchans` default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>spw</b>	string	None	Set of data selection spectral windows, '\'' for all
<b>intent</b>	string	None	Data selection intent: default is first of BANDPASS, PHASE or AMPLITUDE to have data
<b>flag_hilo</b>	bool	True	True to flag outlier baseline/channels
<b>fhl_limit</b>	double	20.0	Flag baseline/channels further from median than limit * MAD
<b>fhl_minsample</b>	double	5	Minimum number of points in sample
<b>flag_bad_quadrant</b>	bool	True	True to flag bad quadrants in antennas
<b>fbq_hilo_limit</b>	double	8.0	Note as '\''suspect'\'' baseline/channels further from median than limit * MAD
<b>fbq_antenna_frac_limit</b>	double	0.2	Flag antenna quadrant if its fraction of '\''suspect'\'' baseline/channels > limit
<b>fbq_baseline_frac_limit</b>	double	1.0	Flag baseline quadrant if its fraction of '\''suspect'\'' baseline/channels > limit

<b>pipelinemode</b>	string	automatic	The pipeline operations mode
<b>dryrun</b>	bool	False	Run the task (False) or list commands(True)
<b>acceptresults</b>	bool	True	Automatically apply results to context

## 6.21 hif\_refant

The hif\_refant task selects a list of reference antennas and stores them in the pipeline context in priority order. The priority order is determined by a weighted combination of scores derived by the antenna selection heuristics. In manual mode the reference antennas can be set by hand.

### Task Description

Select the best reference antennas

The hif\_refant task selects a list of reference antennas and outputs them in priority order. The priority order is determined by a weighted combination of scores derived by the antenna selection heuristics.

### Keyword arguments:

#### ---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

**hm\_refant** -- The heuristics method or mode for selection the reference antenna. The options are 'manual' and 'automatic'. In manual mode a user supplied reference antenna refant is supplied. In 'automatic' mode the antennas are selected automatically.  
default: 'automatic'

**refant** -- The user supplied reference antenna for 'manual' mode. If no antenna list is supplied an empty list is returned.

default: ''

example: 'DV05'

**geometry** -- Score antenna by proximity to the center of the array. This option is quick as only the ANTENNA table must be read.

default: True

**flagging** -- Score antennas by percentage of unflagged data. This option requires computing flagging statistics.

default: True

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets in the pipeline context.

default: ''

example: ['M31.ms']

**field** -- The list of field names or field ids for which flagging scores are computed if hm\_refant='automatic' and flagging = True  
default: ""  
example: '3C279', '3C279', M82'

**intent** -- A string containing a comma delimited list of intents against which the selected fields are matched. Defaults to all supported intents.  
default: ""  
example: '\*BANDPASS\*'

**spw** -- The list of spectral windows and channels for which flagging scores are computed if hm\_refant='automatic' and flagging = True.  
default: ""  
example: '11,13,15,17'

### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: True  
**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Examples

1. Compute the references antennas to be used for bandpass and gain calibration.  
hif\_refant()

### Parameter List

**Table 26: hif\_refant default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>field</b>	string	None	Set of data selection field names or ids
<b>spw</b>	string	None	Set of data selection spectral windows / channels
<b>intent</b>	string	None	Set of data selection intents
<b>hm_refant</b>	string	automatic	The reference antenna heuristics mode
<b>refant</b>	string	None	List of reference antennas
<b>geometry</b>	bool	True	Score by proximity to

			center of the array
<b>flagging</b>	bool	True	Score by percentage of good data
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Add the results into the pipeline context

## 6.22 *hif\_restoredata*

The *hif\_restoredata* restores flagged and calibrated data from archived ASDMs and pipeline flagging and calibration data products. Pending archive retrieval support *hif\_restore data* assumes that the required products are available in the *rawdata\_dir* in the format produced by the *hif\_exportdata* task. *hif\_restoredata* assumes that the following entities are available in the raw data directory

- o the ASDMs to be restored
- o for each ASDM in the input list
- o a compressed tar file of the final flagversions file, e.g.  
uid\_\_\_\_A002\_X30a93d\_X43e.ms.flagversions.tar.gz
- o a text file containing the applycal instructions, e.g.  
uid\_\_\_\_A002\_X30a93d\_X43e.ms.calapply.txt
- o a compressed tar file containing the caltables for the parent session,  
e.g. uid\_\_\_\_A001\_X74\_X29.session\_3.caltables.tar.gz
- hif\_restore data performs the followinga operations
- o imports the ASDM(s)
- o removes the default MS.flagversions directory created by the filler
- o restores the final MS.flagversions directory stored by the pipeline
- o restores the final set of pipeine flags to the MS
- o restores the final calibration state of the MS
- o restores the final calibration tables for each MS
- o applies the calibration tables to each MS

### Task Description

Restore flagged and calibration interferometry data from a pipeline run The *hif\_restoredata* task restores flagged and calibrated measurements sets from archived ASDMs and pipeline flagging and calibration date products.

### Keyword arguments:

#### ---- pipeline parameters that can be set in any *pipelinemode*

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In 'interactive' mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task. default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- List of raw visibility data files to be restored. Assumed to be in the directory specified by rawdata\_dir.

default: None

example: vis=['uid\_\_\_\_A002\_X30a93d\_X43e']

**session** -- List of sessions one per visibility file.

default: None

example: session=['session\_3']

**products\_dir** -- Name of the data products directory. Currently not used.

default: '../products'

example: products\_dir='myproductspath'

**rawdata\_dir** -- Name of the rawdata subdirectory.

default: '../rawdata'

example: rawdata\_dir='myrawdatopath'

### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Examples

1. Restore the pipeline results for a single ASDM in a single session

hif\_restoredata (vis=['uid\_\_\_\_A002\_X30a93d\_X43e'], session=['session\_1'])

### Parameter List

**Table 27: hif\_restoredata default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input visibility data
<b>session</b>	stringArray	None	List of sessions one per visibility file
<b>products_dir</b>	string	../products	The archived pipeline data products directory
<b>copytoraw</b>	bool	True	Copy calibration and flagging tables to raw data directory
<b>rawdata_dir</b>	string	../rawdata	The rawdata directory
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display task

			command (True)
<b>acceptresults</b>	bool	True	Add the results into the pipeline context

## 6.23 *hif\_setjy*

Fills the model column with the model visibilities

### Task Description

Fill the model column with calibrated visibilities

Fills the model column with the model visibilities.

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.  
default: 'automatic'.

### ---- pipeline parameters that can be set in any pipelinemode

**fluxdensity** -- Specified flux density [I,Q,U,V] in Jy. Uses [1,0,0,0] flux density for unrecognized sources, and standard flux densities for ones recognized by 'standard', including 3C286, 3C48, 3C147, and several planets, moons, and asteroids.  
default=-1  
example: [3.06,0.0,0.0,0.0]

**reffile** -- Path to a file containing flux densities for calibrators unknown to CASA. Values given in this file take precedence over the CASA-derived values for all calibrators except solar system calibrators. By default the path is set to the CSV file created by h\_importdata, consisting of catalogue fluxes extracted from the ASDM.  
default: "  
example: "", 'working/flux.csv'

**spix** -- Spectral index for fluxdensity  $S = \text{fluxdensity} * (\text{freq}/\text{reffreq})^{\text{spix}}$   
Only used if fluxdensity is being used. If fluxdensity is positive, and spix is nonzero, then reffreq must be set too. It is applied in the same way to all polarizations, and does not account for Faraday rotation or depolarization.  
Default: 0

**reffreq** -- The reference frequency for spix, given with units. Provided to avoid division by zero. If the flux density is being scaled by spectral index, then reffreq must be set to whatever reference frequency is correct for the given fluxdensity and spix. It cannot be determined from vis. On the other hand, if spix is 0, then any positive frequency can be used and will be ignored.  
Default: '1GHz'  
Examples: '86.0GHz', '4.65e9Hz'

**scalebychan** -- This determines whether the fluxdensity set in the model is calculated on a per channel basis. If False then only one fluxdensity

value is calculated per spw.

default: True

**standard** -- Flux density standard, used if fluxdensity[0] less than 0.0. The options are: 'Baars', 'Perley 90', 'Perley-Taylor 95', 'Perley-Taylor 99', 'Perley-Butler 2010' and 'Butler-JPL-Horizons 2010'.

default: 'Butler-JPL-Horizons 2012' for solar system object

'Perley-Butler 2010' otherwise

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets defined in the pipeline context.

default: []

example:

**field** -- The list of field names or field ids for which the models are to be set. Defaults to all fields with intent **\*AMPLITUDE\***.

default: ''

example: '3C279', '3C279', M82'

**intent** -- A string containing a comma delimited list of intents against which the selected fields are matched. Defaults to all data with amplitude intent.

default: ''

example: **\*AMPLITUDE\***

**spw** -- The list of spectral windows and channels for which bandpasses are computed. Defaults to all science spectral windows.

default: ''

example: '11,13,15,17'

**model** -- Model image for setting model visibilities. Not fully supported.

default: ''

example: see details in help for CASA setjy task

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

#### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

#### Examples

1. Set the model flux densities for all the amplitude calibrators.  
hif\_setjy()

## Parameter List

**Table 28: hif\_setjy default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>field</b>	string	None	List of field names or ids
<b>intent</b>	string	None	Observing intent of flux calibrators
<b>spw</b>	string	None	List of spectral window ids
<b>model</b>	string	None	File location for field model
<b>reffile</b>	string	None	Path to file with fluxes for non-solar system calibrators
<b>fluxdensity</b>	any	-1	Specified flux density [I,Q,U,V]; -1 will lookup values
<b>spix</b>	double	0.0	Spectral index of fluxdensity
<b>reffreq</b>	string	1GHz	Reference frequency for spix
<b>scalebychan</b>	bool	True	Scale the flux density on a per channel basis or else on a per spw basis
<b>standard</b>	variant	None	Flux density standard
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the commands (True)
<b>acceptresults</b>	bool	True	Automatically accept results into the context

## 6.24 hif\_show\_calstate

hif\_show\_calstate displays the current on-the-fly calibration state of the pipeline as a set of equivalent applycal calls.

### Task Description

Show the current pipeline calibration state

### Keyword arguments:

None

## Parameter List

No parameters

## **6.25 hif\_tclean**

### **Task Description**

Compute clean map Compute a cleaned image for a particular target source/intent and spectral window.

#### **Keyword arguments:**

##### **--- pipeline parameters that can be set in any pipelinemode**

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.

default: 'automatic'.

##### **---- pipeline-defined parameters which can be set only in 'interactive mode'**

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets in the context. default: " example: vis=['ngc5921a.ms', ngc5921b.ms', 'ngc5921c.ms']

**imagename** -- Prefix of output images. Defaults to one of the following options depending on the availability of project information. '{ousstatus uid}.{field}.[{intent}].s{stage number}.spw{spw}' 'multivis.{field}.[{intent}].s{stage number}.spw{spw}' cleanboxes and thresholds to use as it goes. For each iteration the output images are:

{prename}.iter{n}.image; cleaned and restored image {prename}.iter{n}.psf; point spread function (dirty beam) {prename}.iter{n}.flux; relative sky sensitivity over field {prename}.iter{n}.flux.pbcoverage; relative pb coverage over field

(only for mosaics) {prename}.iter{n}.model; image of clean components

{prename}.iter{n}.residual; image of residuals

{prename}.iter{n}.cleanmask; image of cleanmask used default: " example: 'test1'

**intent** -- An intent against which the selected fields are matched. Default means select all data from fields specified by 'field' parameter default: " example: "", 'TARGET'

**field** -- Fields id(s) or name(s) to image or mosaic. Must be set. default:

example: '3C279', 'Centaurus\*'

**spw** -- Spectral window/channels to image. '\\" for all science data. default: "

example: '9', '9,11' spwsel -- Spectral window frequency selection for continuum. {} for all science data.

default: {}

example: {'spw16': '89.1~89.5GHz;90.2~90.3GHz', 'spw18': '101.2~102.1GHz'} specmode -- Frequency imaging mode, 'mfs', 'cube'. '\\" defaults to

'cube' if intent parameter includes 'TARGET' otherwise 'mfs'. default: " example: 'mfs', 'cube'

**gridder** -- Gridding options, 'standard', 'mosaic'. Derived as follows:

1. The 'field' parameter is converted into a list of field\_ids for each measurement set in 'vis'.
2. If there is more than 1 field\_id in the list for any measurement set then gridder is set to 'mosaic', otherwise it will be set to 'standard'.

default: ''

'standard' deconvolver -- Minor cycle algorithm e.g. hogbom or clark clean. '\' defaults to 'hogbom' outframe -- The reference frame of the output image. The only supported option is 'LSRK' default: '' example: 'LSRK'

**imsize** -- X and Y image size in pixels). Must be even and contain factors 2,3,5,7 only. Default derived as follows:

1. Determine 'phasecenter' value and spread of field centres around it.
  2. Set size of image to cover spread of field centres plus a border of width  $0.75 * \text{beam radius}$  (to first null).
  3. Divide x and y extents by 'cell' values to arrive at the numbers of pixels required.
- default: ''

example: [320,320] cell -- X and Y cell size. Derived from maximum UV spacing. Details TBD  
default ''

example: ['0.5arcsec', '0.5arcsec'] phasecenter -- Direction measure or field id for the mosaic center.

Default derived as follows: 1. Make an array containing all the field centers to be imaged together. 2. Derive the mean direction from the directions array.

default: ''

example: 2 nchan -- Number of channels or planes in the output image, -1 for all

default: -1 example: 128

**width** -- Width of spectral dimension in frequency, '\' for default. default: '' example: '7.8125MHz'

weighting -- Weighting to apply to visibilities. Options are: 'natural', 'uniform','briggs', 'superuniform','briggsabs','radial'

default='briggs'

example: weighting='uniform' robust -- Parameter for 'briggs' and 'briggsabs' weighting. Ranges from -2.0 to

2.0. -2 for uniform +2 for natural. default=-999.0 example: 0.7

**noise** -- Parameter for 'briggsabs' weighting default: '1.0Jy' example: '0.5Jy'

**npixels** -- Parameter for 'briggs' and 'briggsabs'; weighting default: 0

example: 0 **restoringbeam** -- Gaussina sestoring beam for clean, '\\" for default

default: '\\"

example: **hm\_masking** -- Clean masking mode. Options are 'none', 'centralquarter'

'psf', 'psfilter' and 'manual' default: 'centralquarter' example: 'manual'

**mask** -- Image mask for hm\_masking manual mode. User responsible for matching image sizes, coordinates, etc. default: '\\" example: 'mymask.mask'

**niter** -- Maximum number of iterations per clean call default: 1000 example: 500

**threshold** -- Threshold for cleaning default: '0.0' example: '0.05'

**maxncleans** -- Maximum number of clean calls default: 1

example: 10 **subcontms** -- Subtract continuum in MS

Caveat: if enabled, repeated continuum subtraction will lead to erroneous negative fluxes.

default=False

Options: False, True **parallel** -- use MPI cluster to clean images

default: '\\'automatic\' --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False **acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned. Examples: Make an 'mfs' image of calibrator 3c279 using data in spectral window 1. The cell size is set to 0.2 arcsec in RA and Dec. Other clean parameters are derived from heuristics: hif\_tclean(field='3c279', cell='0.2arcsec', spw='1', specmode='mfs') Make a cube of calibrator 3c279 using data in spectral window 1. The cube planes will be evenly spaced in frequency in the LSRK frame. Other clean parameters are derived from heuristics.

hif\_tclean(field='3c279', cell='0.2arcsec', spw='1', specmode='cube', outframe='LSRK')

## Parameter List

**Table 29: hif\_tclean default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets, '\'' for default
<b>imagename</b>	string	None	Prefix for image filenames, '\'' for default
<b>intent</b>	string	None	Set of data selection intents, '\'' for all
<b>field</b>	string	None	Set of data selection field names or ids
<b>spw</b>	string	None	Set of data selection spectral window/channels '\'' for all
<b>spwsel</b>	any	{}	Dictionary of spw frequency selections {} for automatic
<b>uvrange</b>	any	None	Set of uv ranges, '\'' for all
<b>specmode</b>	string	None	Spectral gridding type (mfs, cube, '\'' for default)
<b>gridder</b>	string	None	Gridding options (standard, mosaic, '\'' for default)
<b>deconvolver</b>	string	None	Minor cycle algorithm (hogbom, clark, mtmfs, '\'' for default)
<b>outframe</b>	string	None	velocity frame of output image (LSRK, '\'' for default)
<b>imsize</b>	intArray	None	X and Y image size in pixels, single value same for both, '\'' for default
<b>cell</b>	stringArray	None	X and Y cell size(s), single value same for both, '\'' for default
<b>phasecenter</b>	any	None	Image center (direction or field index), '\'' for default
<b>nchan</b>	int	-1	Number of channels or planes in output image, -1 = all
<b>start</b>	any	None	Start of output spectral dimension
<b>width</b>	any	None	Width of output spectral channels, '\'' for default
<b>weighting</b>	string	briggs	Type of weighting
<b>robust</b>	double	-999.0	Briggs weighting robustness parameter
<b>noise</b>	any	1.0Jy	Briggs weighting noise parameter
<b>npixels</b>	int	0	Weighting algorithm parameter
<b>restoringbeam</b>	stringArray	None	Gaussian restoring beam, '\'' for default

<b>hm_masking</b>	string	none	Pipeline heuristics masking option
<b>hm_cleaning</b>	string	manual	Pipeline clean control heuristics
<b>mask</b>	any	None	User mask, '\\' for whole image
<b>niter</b>	int	1000	Maximum number of clean iterations
<b>threshold</b>	double	0.0	Flux level to stop cleaning, must include units: '1.0mJy'
<b>tlimit</b>	double	2.0	Times the sensitivity limit for cleaning
<b>masklimit</b>	int	4	Times good mask pixels for cleaning
<b>maxncleans</b>	int	1	Maximum number of clean task calls
<b>subcontms</b>	bool	False	Subtract continuum in MS
<b>parallel</b>	string	automatic	Clean images using MPI cluster
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command(True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## 7 Interferometry ALMA Task Descriptions

### 7.1 *hifa\_antpos*

The hifa\_antpos task corrects the antenna positions recorded in the ASDMs using updated antenna position calibration information determined after the observation was taken. Corrections can be input by hand, read from a file on disk, or in future by querying an ALMA database service. The antenna positions file is in 'csv' format containing 6 comma delimited columns as shown below. The default name of this file is 'antennapos.csv' with the format:

```
file ms,antenna,xoffset,yoffset,zoffset,comment
uid__A002_X30a93d_X43e.ms,DV11,0.000,0.010,0.000,"No comment"
uid__A002_X30a93d_X43e.dup.ms,DV11,0.000,-0.010,0.000,"No comment"
```

The corrections are used to generate a calibration table which is recorded in the pipeline context and applied to the raw visibility data, on the fly to generate other calibration tables, or permanently to generate calibrated visibilities for imaging. Issues The hm\_antpos 'online' option will be implemented when the observing system provides an antenna position determination service.

#### Task Description

Derive an antenna position calibration table

Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context dependent pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.

default: 'automatic'.

**---- pipeline parameters that can be set in any pipelinemode**

**hm\_antpos** -- Heuristics method for retrieving the antenna position corrections. The options are 'online' (not yet implemented), 'manual', and 'file'. default: 'file' example: hm\_antpos='manual'

**antenna** -- The list of antennas for which the positions are to be corrected if hm\_antpos is 'manual' default: none example 'DV05,DV07'

**offsets** -- The list of antenna offsets for each antenna in 'antennas'. Each offset is a set of 3 floating point numbers separated by commas, specified in the ITRF frame. default: none example: [0.01, 0.02, 0.03, 0.03, 0.02, 0.01]

**antposfile** -- The file(s) containing the antenna offsets. Used if hm\_antpos is 'file'. The default file name is 'antennapos.csv'

**---- pipeline-defined parameters which can be set only in 'interactive mode'**

**vis** -- List of input visibility files

default: []

example: ['ngc5921.ms'] **citable** -- Name of output gain calibration tables

default: []

example: citable=['ngc5921.gcal']

**-- Pipeline task execution modes**

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True **acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True **Output: results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

**Examples**

1. Correct the position of antenna 5 for all the visibility files in a single pipeline run.

hifa\_antpos (antenna='DV05', offsets=[0.01, 0.02, 0.03])

2. Correct the position of antennas for all the visibility files in a single pipeline run using antenna positions files on disk. These files are assumed to conform to a default naming scheme if 'antposfile' is unspecified by the user.

`hifa_antpos (hm_antpos='myantposfile.csv')`

### Parameter List

**Table 30: hifa\_antpos default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>caltable</b>	stringArray	None	List of output caltable(s)
<b>hm_antpos</b>	string	file	The antenna position determination method
<b>antenna</b>	string	None	List of antennas to be corrected
<b>offsets</b>	doubleArray	None	List of position corrections one set per antenna
<b>antposfile</b>	string	None	File containing antenna position corrections
<b>pipelinemode</b>	string	automatic	The pipeline operation mode
<b>dryrun</b>	bool	False	Run the task (False) or list commands(True)
<b>acceptresults</b>	bool	True	Automatically accept results into context

## 7.2 *hifa\_bandpass*

`hifa_bandpass` computes a bandpass solution for every specified science spectral window. By default a 'phaseup' pre-calibration is performed and applied on the fly to the data, before the bandpass is computed. `hifa_bandpass` supercedes `hif_bandpass` for ALMA data.

### Task Description

Compute bandpass calibration solutions

Compute amplitude and phase as a function of frequency for each spectral window in each measurement set.

Previous calibration can be applied on the fly.

### Keyword arguments:

#### --- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically.

In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.  
default: 'automatic'.

**hm\_phaseup** -- The pre-bandpass solution phaseup gain heuristics. The options are 'snr' (compute solution required to achieve the specified SNR), 'manual' (use manual solution parameters), and " (none).

default: 'snr'

example: hm\_phaseup='manual'

**phaseupsolint** -- The phase correction solution interval in CASA syntax. Used when hm\_phaseup='manual' or as a default if the hm\_phasup='snr' heuristic computation fails.

default: 'int'

example: phaseupsolint='300s'

**phaseupbw** -- Bandwidth to be used for phaseup. Defaults to 500MHz.

Used when hm\_phaseup='manual'.

default: "

example: " default to entire bandpass, '500MHz' use centreal 500MHz

**phaseupsnr** -- The required SNR for the phaseup solution. Used only if hm\_phaseup='snr'

default: 20.0

example: phaseupsnr=10.0

**phaseupnsols** -- The minimum number of phaseup gain solutions. Used only if hm\_phaseup='snr'.

default: 2

example: phaseupnsols=4

**hm\_bandpass** -- The bandpass solution heuristics. The options are 'snr' (compute the solution required to achieve the specified SNR), 'smoothed' (simple smoothing heuristics), and 'fixed' (use the user defined parameters for all spws).

**solint** -- Time and channel solution intervals in CASA syntax.

default: 'inf' Used for hm\_bandpass='fixed', and as a default for the 'snr' and 'smoothed' options.

default: 'inf,7.8125MHz'

example: solint='inf,10ch', solint='inf'

**maxchannels** -- The bandpass solution smoothing factor in channels. The solution interval is bandwidth / 240. Set to 0 for no smoothing.

Used if hm\_bandpass='smoothed'.

default: 240

example: 0

**bpsnr** -- The required SNR for the bandpass solution. Used only if hm\_bandpass='snr'

default: 50.0

example: bpsnr=20.0

**bpnsols** -- The minimum number of bandpass solutions. Used only if hm\_bandpass='snr'.

default: 8

**hm\_bandtype** -- The type of bandpass. The options are 'channel' and 'polynomial' for CASA bandpass types = 'B' and 'BPOLY' respectively.

**combine** -- Data axes to combine for solving. Axes are "", 'scan', 'spw', 'field' or any comma-separated combination.

default: 'scan'  
example: combine='scan,field'

**minblperant** -- Minimum number of baselines required per antenna for each solve  
Antennas with fewer baselines are excluded from solutions. Used for  
hm\_bandtype='channel' only.  
default: 4

**minsnr** -- Solutions below this SNR are rejected. Used for hm\_bandtype=  
'channel' only  
default: 3.0

**---- pipeline-defined parameters which can be set only in  
'interactive mode'**

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the pipeline context.  
default: ""  
example: ['M51.ms']

**caltable** -- The list of output calibration tables. Defaults to the standard pipeline naming convention.  
default: ""  
example: ['M51.bcal']

**field** -- The list of field names or field ids for which bandpasses are computed. Defaults to all fields.  
default: ""  
example: '3C279', '3C279, M82'

**intent** -- A string containing a comma delimited list of intents against which the selected fields are matched. Defaults to all data with bandpass intent.  
default: ""  
example: '\*PHASE\*'

**spw** -- The list of spectral windows and channels for which bandpasses are computed. Defaults to all science spectral windows.  
default: ""  
example: '11,13,15,17'

**refant** -- Reference antenna names. Defaults to the value(s) stored in the pipeline context. If undefined in the pipeline context defaults to the CASA reference antenna naming scheme.  
default: ""  
example: refant='DV01', refant='DV06,DV07'

**solnorm** -- Normalise the bandpass solutions  
default: False

**--- pipeline task execution modes**

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Examples

1. Compute a channel bandpass for all visibility files in the pipeline context using the CASA reference antenna determination scheme.

hif\_bandpass()

2. Same as the above but precompute a prioritized reference antenna list

hif\_refant()

hif\_bandpass()

### Parameter List

**Table 31: hif\_bandpass default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>caltable</b>	stringArray	None	List of output caltables
<b>field</b>	string	None	Set of data selection field names or ids
<b>intent</b>	string	None	Set of data selection intents
<b>spw</b>	string	None	Set of data selection spectral windows/channels
<b>antenna</b>	string	None	Set of data selection antenna IDs
<b>hm_phaseup</b>	string	snr	Phaseup before computing the bandpass
<b>phaseupsolint</b>	any	int	Phaseup correction solution interval
<b>phaseupbw</b>	string	None	Bandwidth to use for phaseup
<b>phaseupsnr</b>	double	20.0	SNR for phaseup solution
<b>phaseupsols</b>	int	2	Minimum number of phaseup gain solutions
<b>hm_bandpass</b>	string	snr	Bandpass solution heuristics
<b>solint</b>	any	inf	Solution intervals
<b>maxchannels</b>	int	240	The smoothing factor in channels
<b>bpsnr</b>	double	50.0	SNR for bandpass solution
<b>bponsols</b>	int	8	Minimum number of bandpass solutions

<b>hm_bandtype</b>	string	channel	Bandpass solution type
<b>combine</b>	string	scan	Data axes to combine for solve (scan, spw, and/or field)
<b>refant</b>	string	None	Reference antenna names
<b>solnorm</b>	bool	True	Normalise the bandpass solution
<b>minblperant</b>	int	4	Minimum baselines per antenna required for solve
<b>minsnr</b>	double	3.0	Reject solutions below this SNR
<b>degamp</b>	variant	None	Degree for polynomial amplitude solution
<b>degphase</b>	variant	None	Degree for polynomial phase solution
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Add the results to the Pipeline context

## 7.3 *hifa\_bpsolint*

### Task Description

Compute optimal bandpass calibration solution intervals

Compute the bandpass phaseup time solution interval and bandpass frequency solution interval.

### ---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline context defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

**phaseupsnr** -- The required phaseup gain time interval solution signal to noise  
default: 20.0

example: phaseupsnr = 10.0

**minphaseupints** -- The minimum number of time intervals in the phaseup gain solution.  
default: 2

example: minphaseupints=4

**bpnsr** -- The required bandpass frequency interval solution signal to noise  
default: 50.0

example: phaseupsnr = 20.0

**minbpnchan** -- The minimum number of frequency intervals in the bandpass solution.  
default: 8  
example: minbpnchans=16

**hm\_nantennas** -- The heuristics for determines the number of antennas to use in the signal to noise estimate. The options are 'all' and 'unflagged'.  
The 'unflagged' option is not currently supported.  
default: 'all'  
example: hm\_nantennas='unflagged'

**maxfracflagged** -- The maximum fraction of an antenna that can be flagged before it is excluded from the signal to noise estimate.  
default: 0.90  
example: maxfracflagged=0.80

#### **---- pipeline-defined parameters which can be set only in 'interactive mode'**

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the pipeline context  
default: ""  
example: ['M82A.ms', 'M82B.ms']

**field** -- The list of field names of sources to be used for signal to noise estimation. Defaults to all fields with the standard intent.  
default: ""  
example: '3C279'

**intent** -- A string the intent against which the selected fields are matched.  
Defaults to 'BANDPASS'.  
default: 'BANDPASS'  
example: intent='PHASE'

**spw** -- The list of spectral windows and channels for which gain solutions are computed. Defaults to all the science spectral windows for which there are both 'intent' and TARGET intents.  
default: ""  
example: '13,15'

#### **--- pipeline task execution modes**

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: False  
**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

#### **Output:**

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned

#### **Description**

The phaseup gain time and bandpass frequency intervals are determined as follows.

- o For each data set the list of source(s) to use for bandpass solution signal to noise estimation is compiled based on the values of the field, intent, and spw parameters.
- o Source fluxes are determined for each spw and source combination.
- o Fluxes in Jy are derived from the pipeline context.
- o Pipeline context fluxes are derived from the online flux calibrator catalog, the ASDM, or the user via the flux.csv file.
- o If no fluxes are available the task terminates.
- o Atmospheric calibration and observations scans are determined for each spw and source combination.
- o If intent is set to 'PHASE' and there are no atmospheric scans associated with the 'PHASE' calibrator, 'TARGET' atmospheric scans will be used instead.
- o If atmospheric scans cannot be associated with any of the spw and source combinations the task terminates.
- o Science spws are mapped to atmospheric spws for each science spw and source combinations.
- o If mappings cannot be determined for any of the spws the task terminates.
- o The median Tsys value for each atmospheric spw and source combination is determined from the SYSCAL table. Medians are computed first by channel, then by antenna, in order to reduce sensitivity to deviant values.
- o The science spw parameters, exposure time(s), and integration time(s) are determined.
- o The phase up time interval, in time units and number of integrations required to meet the phaseupsnr are computed, along with the phaseup sensitivity in mJy and the signal to noise per integration. Nominal Tsys and sensitivity values per receiver band provide by the ALMA project are used for this estimate.
- o Warnings are issued if estimated phaseup gain time solution would contain fewer than minphasupints solutions
- o The frequency interval, in MHz and number of channels required to meet the bpsnr are computed, along with the per channel sensitivity in mJy and the per channel signal to noise. Nominal Tsys and sensitivity values per receiver band provide by the ALMA project are used for this estimate.
- o Warnings are issued if estimated bandpass solution would contain fewer than minbpnchan solutions

## Examples

1. Estimate the phaseup gain time interval and the bandpass frequency interval required to match the desired signal to noise for bandpass solutions.  
hifa\_bpsolint()

## Parameter List

**Table 32: hifa\_bpsolint default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>field</b>	string	None	Set of data selection field names

<b>intent</b>	string	BANDPASS	Set of data selection observing intents
<b>spw</b>	string	None	Set of data selection spectral window IDs
<b>phaseupsnr</b>	double	20.0	The required bandpass phaseup signal to noise
<b>minphaseupints</b>	int	2	The minimum number of phaseup intervals in the time solution
<b>bpsnr</b>	double	50.0	The required bandpass frequency signal to noise
<b>minbpchan</b>	int	8	The minimum number of channels in the frequency solution
<b>hm_nantennas</b>	bool	all	The antenna selection heuristic (unsupported)
<b>maxfracflagged</b>	string	0.90	The maximum fraction of data flagged per antenna (unsupported)
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Automatically accept the results into the context

## 7.4 hifa\_flagdata

The hifa\_flagdata data performs basic flagging operations on a list of measurements including:

- o applying online flags
- o apply a flagging template (manual flagging provided by user)
- o autocorrelation data flagging
- o shadowed antenna data flagging
- o scan based flagging by intent or scan number
- o edge channel flagging

### Task Description

Do basic flagging of a list of measurement sets

The hifa\_flagdata data performs basic flagging operations on a list of measurement sets.

### Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically.  
In interactive mode the user can set the pipeline-defined parameters

manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.  
default: 'automatic'.

#### ---- pipeline parameters that can be set in any pipelinemode

**autocorr** -- Flag autocorrelation data.  
default: True

**shadow** -- Flag shadowed antennas.  
default: True

**scan** -- Flag a list of specified scans.  
default: True

**scannumber** -- A string containing a comma delimited list of scans to be flagged.  
example: '3,5,6'  
default: "

**intents** -- A string containing a comma delimited list of intents against which the scans to be flagged are matched.  
example: '\*BANDPASS\*'  
default: 'POINTING,FOCUS,ATMOSPHERE,SIDEband'

**edgespw** -- Flag the edge spectral window channels.  
default: True

**fracspw** -- Fraction of the baseline correlator TDM edge channels to be flagged.  
default: 0.0625

**fracspwfps** -- Fraction of the ACS correlator TDM edge channels to be flagged.  
default: 0.48387  
**online** -- Apply the online flags.  
default: True

**fileonline** -- File containing the online flags. These are computed by the h\_init or hif\_importdata data tasks. If the online flags files are undefined a name of the form 'msname\_flagonline.txt' is assumed.  
default: "  
**template** -- Apply flagging templates  
default: True

**filetemplate** -- The name of an text file that contains the flagging template for RFI, birdies, telluric lines, etc. If the template flags files is undefined a name of the form 'msname\_flagtemplate.txt' is assumed.  
default: "

**hm\_tbuff** -- The heuristic for computing the default time interval padding parameter. The options are 'halfint' and 'manual'. In 'halfint' mode tbuff is set to half the maximum of the median integration time of the science and calibrator target observations.  
default: 'halfint'

**tbuff** -- The time in seconds used to pad flagging command time intervals if hm\_tbuff='manual'.  
default: 0.0

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement defined in the pipeline context.

example:

default: "

**flagbackup** -- Back up any pre-existing flags.

default: False

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

#### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

#### Examples

1. Do basic flagging on a measurement set

hifa\_flagdata()

2. Do basic flagging on a measurement set flaggin additional scans selected by number as well.

hifa\_flagdata(scannumber='13,18')

#### Parameter List

Table 33: hifa\_flagdata default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets to flag
<b>autocorr</b>	bool	True	Flag autocorrelation data
<b>shadow</b>	bool	True	Flag shadowed antennas
<b>scan</b>	bool	True	Flag specified scans
<b>scannumber</b>	string	None	List of scans to be flagged
<b>intents</b>	string	POINTING,FOCUS,ATMOSPHERE,SIDEband	List of intents of scans to be flagged

<b>edgespw</b>	bool	True	Flag edge channels
<b>fracspw</b>	double	0.0625	Fraction of baseline correlator edge channels to be flagged
<b>fracspwfps</b>	double	0.048387	Fraction of ACA correlator edge channels to be flagged
<b>online</b>	bool	True	Apply the online flags
<b>fileonline</b>	string	None	File of online flags to be applied
<b>template</b>	bool	True	Apply a flagging template
<b>filetemplate</b>	stringArray	None	File that contains the flagging template
<b>hm_tbuff</b>	string	halfint	The time buffer computation heuristic
<b>tbuff</b>	any	0.0	List of time buffers (sec) to pad timerange in flag commands
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>flagbackup</b>	bool	False	Backup preexisting flags before applying new ones
<b>dryrun</b>	bool	False	Run the task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Add the results into the pipeline context

## 7.5 *hifa\_fluxcalflag*

Search the builtin solar system flux calibrator line catalog for overlaps with the science spectral windows. Generate a list of line overlap regions and flagging commands.

## Task Description

Locate line regions in solar system flux calibrator spws  
Fills the model column with the model visibilities.

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.  
default: 'automatic'.

### ---- pipeline parameters that can be set in any pipelinemode

**threshold** -- If the fraction of an spw occupied by line regions is greater than threshold flag the entire spectral window.

### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets defined in the pipeline context.  
default: []  
example:

**field** -- The list of field names or field ids for which the models are to be set. Defaults to all fields with intent 'AMPLITUDE'.  
default: ""  
example: '3C279', '3C279', M82'

**intent** -- A string containing a comma delimited list of intents against which the selected fields are matched. Defaults to all data with amplitude intent.  
default: ""  
example: 'AMPLITUDE'

**spw** -- The list of spectral windows and channels for which bandpasses are computed. Defaults to all science spectral windows.  
default: ""  
example: '11,13,15,17'

### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

## Examples

1. Locate known lines in any solar system object flux calibrators.  
`hifa_fluxcalflag()`

## Parameter List

Table 34: `hifa_fluxcalflag` default settings

Parameter	Type	Default	Description
<code>vis</code>	stringArray	None	List of input measurement sets
<code>field</code>	string	None	List of field names or ids
<code>intent</code>	string	None	Observing intent of flux calibrators
<code>spw</code>	string	None	List of spectral window ids
<code>pipelinemode</code>	string	automatic	The pipeline operating mode
<code>threshold</code>	double	0.75	Threshold for flagging the entire spw
<code>appendlines</code>	bool	False	Append user defined line regions to the line dictionary
<code>linesfile</code>	string	None	File containing user defined lines
<code>applyflags</code>	bool	True	Apply the computed flag commands
<code>dryrun</code>	bool	False	Run the task (False) or display the commands(True)
<code>acceptresults</code>	bool	True	Automatically accept results into the context

## 7.6 `hifa_fluxdb`

The task `hifa_fluxdb` connects to the ALMA flux calibrator database.

### Task Description

Connect to flux calibrator database

Connect to the ALMA flux calibrator database

### Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline context defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

**vis** -- List of input visibility files  
default: none; example: vis='ngc5921.ms'

#### -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

#### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

#### Example

1. Connect to the ALMA flux calibrator database  
hifa\_fluxdb()

#### Parameter List

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>pipelinemode</b>	string	automatic	The pipeline operations mode
<b>dryrun</b>	bool	False	Run the task (False) or list commands (True)
<b>acceptresults</b>	bool	True	Automatically apply results to context

## 7.7 hifa\_gfluxscale

Derive flux densities for point source transfer calibrators using flux models for reference calibrators. Flux values are determined by:

- o computing complex gain phase only solutions for all the science spectral windows using the calibrator data selected by the 'reference' and 'refintent' parameters and the 'transfer' and 'transintent' parameters, and the value of the 'phaseupsolint' parameter.
- o computing complex amplitude only solutions for all the science spectral windows using calibrator data selected with 'reference' and 'refintent' parameters and the 'transfer' and 'transintent' parameters, the value of the 'solint' parameter.
- o transferring the flux scale from the reference calibrators to the transfer calibrators using refspwmap for windows without data in the reference calibrators
- o extracted the computed flux values from the CASA logs and inserting them into the MODEL\_DATA column.

Resolved calibrators are handled via antenna selection either automatically,

**hm\_resolvedcals**=’automatic’ or manually, **hm\_resolvedcals**=’manual’. In the former case antennas closer to the reference antenna than the uv distance where visibilities fall to ’peak\_fraction’ of the peak are used. In manual mode the antennas specified in ’antenna’ are used. Note that the flux corrected calibration table computed internally is not currently used in later pipeline apply calibration steps.

## Task Description

Derive flux density scales from standard calibrators

Derive flux densities for point source transfer calibrators using flux models for reference calibrators.

### ---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In ’automatic’ mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In ’getinputs’ mode the users can check the settings of all parameters in the task without running the task.  
default: ’automatic’.

**phaseupsolint** -- Time solution intervals in CASA syntax for the phase solution.  
default: ’int’

example: ’inf’, ’int’, ’100sec’

**solint** -- Time solution intervals in CASA syntax for the amplitude solution.

default: ’inf’

example: ’inf’, ’int’, ’100sec’

**minsnr** -- Minimum signal to noise ratio for gain calibration solutions.

default: 2.0

example: 1.5, 0.0

**hm\_resolvedcals** - Heuristics method for handling resolved calibrators. The options are ’automatic’ and ’manual’. In automatic mode antennas closer to the reference antenna than the uv distance where visibilities fall to ’peak\_fraction’ of the peak are used. In manual mode the antennas specified in ’antenna’ are used.

**antenna** -- A comma delimited string specifying the antenna names or ids to be used for the fluxscale determination. Used in **hm\_resolvedcals**=’manual’ mode.  
default: ”.

example: ’DV16,DV07,DA12,DA08’

**peak\_fraction** -- The limiting UV distance from the reference antenna for antennas to be included in the flux calibration. Defined as the point where the calibrator visibilities have fallen to ’peak\_fraction’ of the peak value.

### ---- pipeline-defined parameters which can be set only in ’interactive mode’

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the pipeline context  
default: ”

example: [’M32A.ms’, ’M32B.ms’]

**reference** -- A string containing a comma delimited list of field names defining the reference calibrators. Defaults to field names with

intent '\*AMP\*'.  
default: ""  
example: 'M82,3C273'

**transfer** -- A string containing a comma delimited list of field names defining the transfer calibrators. Defaults to field names with intent '\*PHASE\*'.  
default: ""  
example: 'J1328+041,J1206+30'

**refintent** -- A string containing a comma delimited list of intents used to select the reference calibrators. Defaults to \*AMP\*.  
default: ""  
example: "", "\*AMP"

**refspwmap** -- Vector of spectral window ids enabling scaling across spectral windows. Defaults to no scaling  
default: [-1]  
example: [1,1,3,3] (4 spws, reference fields in 1 and 3, transfer fields in 0,1,2,3)

**transintent** -- A string containing a comma delimited list of intents defining the transfer calibrators. Defaults to \*PHASE\*.  
default: ""  
example: "", "\*PHASE"

**refant** -- A string specifying the reference antenna(s). By default this is read from the context.  
default: ""  
example: 'DV05'

### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned

### Examples

1. Compute flux flux values for the phase calibrator using model data from the amplitude calibrator.  
`hifa_gfluxscale ()`

### Parameter List

**Table 35: hifa\_gfluxscale default settings**

Parameter	Type	Default	Description
<b>vis</b>	string	None	List of input measurements sets
<b>reference</b>	variant	None	Reference calibrator field name(s)
<b>transfer</b>	variant	None	Transfer calibrator field name(s)
<b>refintent</b>	string	None	Observing intent of reference fields
<b>transintent</b>	string	None	Observing intent of transfer fields
<b>refspwmap</b>	IntArray	-1	Map accross spectral window boundaries
<b>phaseupsolint</b>	any	int	Phaseup correction solution interval
<b>solint</b>	any	Inf	Amplitude correction solution interval
<b>minsnr</b>	double	2.0	Minimum SNR for gain solutions
<b>refant</b>	string	None	The name or ID of the reference antenna
<b>hm_resolvedcals</b>	string	automatic	The resolved calibrators heuristics method
<b>antenna</b>	string	None	Antennas to be used in fluxscale
<b>peak_fraction</b>	double	0.2	Fraction of peak visibility at uv-distance limit of antennas to be used
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display commands (True)
<b>acceptresults</b>	bool	True	Automatically accept results into context

## 7.8 hifa\_importdata

### Task Description

Imports data into the interferometry pipeline

The hifa\_importdata task loads the specified visibility data into the pipeline context unpacking and / or converting it as necessary.

**Keyword arguments:**

#### ---- pipeline parameters that can be set in any pipelinemode

**vis** -- List of visibility data files. These may be ASDMs, tar files of ASDMs, MSs, or tar files of MSs, If ASDM files are specified, they will be

converted to MS format.

default: []

example: vis=['X227.ms', 'asdms.tar.gz']

**session** -- List of sessions to which the visibility files belong. Defaults to a single session containing all the visibility files, otherwise a session must be assigned to each vis file.

default: []

example: session=['Session\_1', 'Sessions\_2']

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In 'interactive' mode the user can set the pipeline context defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.

default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**asis** -- ASDM tables to convert as is

default: 'Antenna Station Receiver Source CalAtmosphere CalWVR'

example: 'Receiver', ''

**process\_caldevice** -- Ingest the ASDM caldevice table

default: False

example: True

**overwrite** -- Overwrite existing MSs on output.

default: False

**bdfflags** -- Apply BDF flags on line

default: True

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

#### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

#### Examples

1. Load an ASDM list in the .. rawData subdirectory into the context.  
hifa\_importdata (vis=['.. rawData/uid\_A002\_X30a93d\_X43e',  
'.. rawData/uid\_A002\_x30a93d\_X44e'])

2. Load an MS in the current directory into the context.  
hifa\_importdata (vis=[uid\_A002\_X30a93d\_X43e.ms])

3. Load a tarred ASDM in .../rawdata into the context.  
hifa\_importdata (vis=['..../rawdata/uid\_A002\_X30a93d\_X43e.tar.gz'])

4. Check the hif\_importdata inputs, then import the data  
myvislist = ['uid\_A002\_X30a93d\_X43e.ms', 'uid\_A002\_x30a93d\_X44e.ms']  
hifa\_importdata(vis=myvislist, pipelinemode='getinputs')  
hifa\_importdata(vis=myvislist)

5. Load an ASDM but check the results before accepting them into the context.  
results = hifa\_importdata (vis=['uid\_A002\_X30a93d\_X43e.ms'],  
acceptresults=False)  
results.accept()

6. Run in dryrun mode before running for real  
results = hifa\_importdata (vis=['uid\_A002\_X30a93d\_X43e.ms'], dryrun=True)  
results = hifa\_importdata (vis=['uid\_A002\_X30a93d\_X43e.ms'])

## Parameter List

Table 36: hifa\_importdata default settings

Parameter	Type	Default	Description
vis	stringArray	None	List of input visibility data
session	stringArray	None	List of visibility data sessions
pipelinemode	string	automatic	The pipeline operating mode
asis	string	Antenna Station Receiver Source CalAtmosphere CalWVR	ASDM to convert as is
process_caldevice	bool	False	Import the caldevice table from the ASDM
overwrite	bool	False	Overwrite existing files on import
bdfflags	bool	True	Apply BDF flags on import
dryrun	bool	False	Run the task (False) or display task command (True)
acceptresults	bool	True	Add the results into the pipeline context

## 7.9 hifa\_lipolcal

### Task Description

Compute polarization calibration  
Compute a polarization calibration.

### Keyword arguments:

--- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets in the context. CURRENTLY THE LIST MUST CONTAIN 1 MEASUREMENT SET.  
default: "

example: vis=['ngc5921a.ms', ngc5921b.ms', 'ngc5921c.ms']

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

#### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Parameter List

Table37: hifa\_linpocal default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets, '' for default
<b>field</b>	string	None	Set of data selection field names or ids
<b>intent</b>	string	None	Set of data selection intents
<b>g0table</b>	string	None	Name of table holding G0 gain - not accounting for source pol
<b>delaytable</b>	string	None	Name of table holding cross-hand delay
<b>xyf0table</b>	string	None	Name of table holding residual X-Y phase spectrum and source Q and U
<b>g1table</b>	string	None	Name of table holding G1 gain - accounting for source pol
<b>df0table</b>	string	None	Name of table holding

			instrument polarization gain
<b>refant</b>	string	None	Reference antenna names
<b>spw</b>	string	None	Set of data selection spectral window/channels
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## 7.10 hifa\_spwphaseup

The spectral window mapping from narrow to wide spectral windows is computed as follows.

- o construct a list of the bandwidths of all the science spectral windows
- o determine the maximum bandwidth in this list maxbandwidth
- o for each science spectral window with bandwidth less than maxbandwidth
- o construct a list of spectral windows with bandwidths greater than minfracmaxbw \* maxbandwidth
- o select the spectral window in this list whose band center most closely matches the band center of the narr spectral window
- o preferentially match within the same baseband if samebb is True otherwise map the science spectral window to itself
- o Phase offsets per spw are determined by computing a phase only gain calibration on the selected data, normally the high signal to noise bandpass calibration observations, using the solution interval 'inf'.
- o Store the spectral window map and the phase offset calibration table in the pipeline context for use by later tasks.

### Task Description

Compute narrow to wide band spw map and per spw phase offsets  
Compute the gain solutions.

### ---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline context defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

**maxnarrowbw** -- The maximum bandwidth defining narrow spectral windows. Values must be in CASA compatible frequency units.  
default: '300MHz'  
example: combine=""

**minfracmaxbw** -- The minimum fraction of the maximum bandwidth in the set of

spws to use for matching.  
default: 0.8  
example: minfracmaxbw=0.75

**samebb** -- Match within the same baseband if possible ?  
default: True  
example: samebb=False

**combine** -- Data axes to combine for solving. Options are ", 'scan', 'spw', 'field'  
or any comma-separated combination.  
default: "  
example: combine="

**minblperant** -- Minimum number of baselines required per antenna for each solve  
Antennas with fewer baselines are excluded from solutions.  
default: 4  
example: minblperant=2

**minsnr** -- Solutions below this SNR are rejected.  
default: 3.0

---- pipeline-defined parameters which can be set only in  
'interactive mode'

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the pipeline context  
default: "  
example: ['M82A.ms', 'M82B.ms']

**citable** -- The list of output calibration tables. Defaults to the standard pipeline naming convention.  
default: "  
example: ['M82.gcal', 'M82B.gcal']

**field** -- The list of field names or field ids for which gain solutions are to be computed. Defaults to all fields with the standard intent.  
default: "  
example: '3C279', '3C279', M82'

**intent** -- A string containing a comma delimited list of intents against which the selected fields are matched. Defaults to 'BANDPASS'  
default: 'BANDPASS'  
example: intent="", intent='\*PHASE\*'

**spw** -- The list of spectral windows and channels for which gain solutions are computed. Defaults to all the science spectral windows.  
default: "  
example: '13,15'

**refant** -- Reference antenna name(s) in priority order. Defaults to most recent values set in the pipeline context. If no reference antenna is defined in the pipeline context the CASA defaults are used.  
default: "  
example: refant='DV01', refant='DV05,DV07'

--- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

## Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned

## Examples

1. Compute the simple narrow to wide spw map and the per spw phase offsets.  
`hif_spwphaseup()`

## Parameter List

**Table 38: hif\_spwphaseup default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>caltable</b>	stringArray	None	List of output caltables
<b>field</b>	string	None	Set of data selection field names or IDs
<b>intent</b>	string	BANDPASS	Set of data selection observing intents
<b>spw</b>	string	None	Set of data selection spectral window/channels
<b>maxnarrowbw</b>	string	300 MHz	The maximum bandwidth defining narrow spectral windows
<b>minfracmaxbw</b>	double	0.8	The minimum fraction of the maximum bandpass for spw matching
<b>samebb</b>	bool	True	Match within the same baseband if possible
<b>combine</b>	string	None	Data axes which to combine for solve (scan, spw, and/or field)
<b>refant</b>	string	None	Reference antenna names
<b>minblperant</b>	int	4	Minimum baselines per antenna required for solve
<b>minsnr</b>	double	3.0	Reject solutions below this SNR

<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	False Run task (False) or display the command(True)
<b>acceptresults</b>	bool	True	Automatically accept results into the context

## ***7.11 hifa\_timegaincal***

The complex gains are derived from the data column (raw data) divided by the model column (usually set with hif\_setjy). The gains are obtained for the specified solution intervals, spw combination and field combination. One gain solution is computed for the science targets and one for the calibrator targets.

Good candidate reference antennas can be determined using the hif\_refant task. Previous calibrations that have been stored in the pipeline context are applied on the fly. Users can interact with these calibrations via the hif\_export\_calstate and hif\_import\_calstate tasks.

### **Task Description**

Determine temporal gains from calibrator observations  
Compute the gain solutions.

#### **---- pipeline parameters that can be set in any pipelinemode**

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

#### **calsolint -- Time solution interval in CASA syntax for calibrator source solutions.**

default: 'int'  
example: 'inf', 'int', '100sec'

**targetsolint** -- Time solution interval in CASA syntax for target source solutions.  
default: 'inf'  
example: 'inf', 'int', '100sec'

**combine** -- Data axes to combine for solving. Options are ", 'scan', 'spw', 'field'" or any comma-separated combination.  
default: "

example: combine="

**minblperant** -- Minimum number of baselines required per antenna for each solve  
Antennas with fewer baselines are excluded from solutions.

default: 4  
example: minblperant=2

**calminsnr** -- Solutions below this SNR are rejected for calibrator solutions.

default: 2.0

**targetminsnr** -- Solutions below this SNR are rejected for science target solutions.  
default: 3.0

#### **---- pipeline-defined parameters which can be set only in 'interactive mode'**

**vis** -- The list of input measurement sets. Defaults to the list of measurement sets specified in the pipeline context  
default: ""  
example: ['M82A.ms', 'M82B.ms']

**calamptable** -- The list of output diagnostic calibration amplitude tables for the calibration targets.  
Defaults to the standard pipeline naming convention.  
default: ""  
example: ['M82.gacal', 'M82B.gacal']

**calphasetable** -- The list of output calibration phase tables for the calibration targets.  
Defaults to the standard pipeline naming convention.  
default: ""  
example: ['M82.gcal', 'M82B.gcal']

**amptable** -- The list of output calibration amplitude tables for the calibration and science targets.  
Defaults to the standard pipeline naming convention.  
default: ""  
example: ['M82.gcal', 'M82B.gcal']

**targetphasetable** -- The list of output phase calibration tables for the science targets.  
Defaults to the standard pipeline naming convention.  
default: ""  
example: ['M82.gcal', 'M82B.gcal']

**field** -- The list of field names or field ids for which gain solutions are to be computed. Defaults to all fields with the standard intent.  
default: ""  
example: '3C279', '3C279, M82'

**intent** -- A string containing a comma delimited list of intents against which the selected fields are matched. Defaults to the equivalent of 'AMPLITUDE,PHASE,BANDPASS'.  
default: ""  
example: "", 'PHASE'

**spw** -- The list of spectral windows and channels for which gain solutions are computed. Defaults to all science spectral windows.  
default: ""  
example: '3C279', '3C279, M82'

**smodel** -- Point source Stokes parameters for source model (experimental)  
Defaults to using standard MODEL\_DATA column data.  
default: []  
example: [1,0,0,0] (I=1, unpolarized)

**refant** -- Reference antenna name(s) in priority order. Defaults to most recent values set in the pipeline context. If no reference antenna is defined in the pipeline context use the CASA defaults.  
default: ""  
example: refant='DV01', refant='DV05,DV07'

**solnorm** -- Normalise the gain solutions  
default: False

### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned

### Examples

1. Compute standard per scan gain solutions that will be used to calibrate the target.

hifa\_timegaincal()

### Parameter List

Table 23: hifa\_timegaincal default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>calamptable</b>	stringArray	None	List of diagnostic output amplitude caltables for calibrator targets
<b>calphasetable</b>	stringArray	None	List of output phase caltables for calibrator targets
<b>targetphasetable</b>	stringArray	None	List of output phase caltables for science targets
<b>amptable</b>	stringArray	None	List of output amp caltables for science targets
<b>field</b>	string	None	Set of data selection field names or ids
<b>intent</b>	string	None	Set of data selection observing intents
<b>spw</b>	string	None	Set of data selection spectral window/channels
<b>antenna</b>	string	None	Set of data selection antenna ids
<b>calsolint</b>	any	Int	Phase solution interval for calibrator

			sources
<b>targetsolint</b>	any	inf	Phase solution interval for science target sources
<b>combine</b>	string	None	Data axes which to combine for solve (scan, spw, and/or field)
<b>refant</b>	string	None	Reference antenna names
<b>solnorm</b>	bool	False	Normalize average solution amplitudes to 1.0
<b>minblperant</b>	int	4	Minimum baselines per antenna required for solve
<b>calminsnr</b>	double	2.0	Reject solutions below this SNR for calibrator solutions
<b>targetminsnr</b>	double	3.0	Reject solutions below this SNR for science solutions
<b>smodel</b>	doubleArray	None	Point source Stokes parameters for source model
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run task (False) or display the command(True)
<b>acceptresults</b>	bool	True	Automatically accept results into the context

## 7.12 *hifa\_tsyscal*

### Task Description

Derive a Tsys calibration table

Derive the Tsys calibration for list of measurement sets

### Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
 default: 'automatic'.

**chantol** -- The tolerance in channels for mapping atmospheric calibration windows (TDM) to science windows (FDM or TDM)  
 default: 1  
 example: 5

---- pipeline parameters that can be set in any pipelinemode

---- pipeline-defined parameters which can be set only in 'interactive mode'

**vis** -- List of input visibility files  
default: none; example: vis='ngc5921.ms'

**caltable** -- Name of output gain calibration tables  
default: none; example: caltable='ngc5921.gcal'

#### -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

#### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

#### Examples

1. Compute the tsys calibration tables for a list of measurement sets  
hif\_tsyscal()

#### Parameter List

Table 24: hifa\_tsyscal default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets
<b>caltable</b>	stringArray	None	List of output caltable(s)
<b>chantol</b>	int	1	Tsys spectral window map channel tolerance
<b>pipelinemode</b>	string	automatic	The pipeline operations mode
<b>dryrun</b>	bool	False	Run the task (False) or list commands(True)
<b>acceptresults</b>	bool	True	Automatically apply results to context

### 7.13 hifa\_tsysflagchans

Deviant channels in Tsys measurements are detected by analysis of a compressed view of the Tsys results. This view comprises a list of Tsys spectra whose details are determined by the metric parameter.

Flags are generated by running the following rules on each spectrum:

If flag\_edges = True then channels at the band edges are detected and flagged.

If flag\_sharps = True then channels covering sharp spectral features are

flagged and then the flagging is extended in an attempt to cover the lower flanks of the features as well.

## Task Description

Flag deviant channels in system temperature measurements  
hif\_tsysflagchans flags deviant channels in the system temperature calibration table.

### Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically. In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.  
default: 'automatic'.

#### ---- pipeline parameters that can be set in any pipelinemode

**metric** -- Metric to use in calculating the flagging views of the Tsys data.

Possible values are:

'median' - A list of flagging views will be generated, one for each SpW/intentgroup. Each view is the median spectrum from all the member spectra.

'antenna\_median' - A flagging view is generated for each SpW/intentgroup/antenna. Each view is the median spectrum from all the member spectra.

'antenna\_diff' - A flagging view is generated for each SpW/intentgroup/antenna. Each view is the difference between the median of spectra for the antenna/SpW and the median for the SpW as a whole.  
default: 'median'

**intentgroups** -- How data from various intents are to be combined in separate flagging views. Internally the intentgroups are stored in a list of strings but, because of interface limitations, this list must be input as a single string.  
default: "[ATMOSPHERE]"

For each spw the default creates a Tsys median spectrum from all spectra with ATMOSPHERE intent.

**flag\_edges** -- True to flag edges of spectra.  
default True

**edge\_limit** -- The first channels inward from each end of the spectrum where the channel to channel difference falls below 'edge\_limit' times the median across the spectrum are designated as the 'edges'. These channels and those outside them are flagged. If the 'edges' lie more than 1/4 of the way across the spectrum then no flagging is done.  
default 3

**flag\_sharps** -- True to flag channels that cover sharp spectral features.  
Flags sharp feature 'cores' where the channel to channel difference exceeds a given limit, then extends the flagging over the feature flanks until the channel to channel difference

falls below 2 \* the median over the spectrum.

default: False

**sharps\_limit** -- Flag as sharp feature cores those channels bracketing a channel to channel difference > sharps\_limit.

default: 0.05

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**caltable** -- List of input Tsys calibration tables

default: [] - Use the table currently stored in the pipeline context.

example: caltable=['X132.ms.tsys.s2.tbl']

#### -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- This parameter has no effect. The Tsyscal file is already in the pipeline context and is flagged in situ.

#### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

#### Examples

1. Flag the edges of the Tsys measurements in each SpW.

hif\_tsysflagchans()

equivalent to:

hif\_tsysflagchans(flag\_edges=True, edge\_limit=3)

2. Flag the edges of the Tsys measurements in each SpW, and sharp spectral features in the band.

hif\_tsysflagchans(flag\_sharps=True)

equivalent to:

hif\_tsysflagchans(flag\_edges=True, edge\_limit=3, flag\_sharps=True, sharps\_limit=0.05)

3. Flag antenna-based 'glitches':

hif\_tsysflagchans(metric='antenna\_diff', flag\_edges=False, flag\_sharps=True)

equivalent to:

hif\_tsysflagchans(metric='antenna\_diff', flag\_edges=False, flag\_sharps=True, sharps\_limit=0.05)

#### Parameter List

Table 41:hifa\_tsysflagchans default settings

Parameter	Type	Default	Description
<b>caltable</b>	stringArray	None	List of input calsables
<b>metric</b>	string	median	Method used to judge the quality of a Tsys measurement
<b>intentgroups</b>	string	"ATMOSPHERE"	List of groups of

			intents for which views are to be created
<b>flag_edges</b>	bool	True	True to flag edge channels of Tsys spectra
<b>edge_limit</b>	double	3	Candidate edge channels have channel to channel difference > edge_limit * median across spectrum
<b>flag_sharps</b>	bool	False	True to flag channels covering sharp spectral features plus additional channels on their flanks
<b>sharps_limit</b>	double	0.05	Flag channels bracketing a channel to channel difference > sharps_limit
<b>pipelinemode</b>	string	automatic	The pipeline operations mode
<b>dryrun</b>	bool	False	Run the task (False) or list commands (True)
<b>acceptresults</b>	bool	True	Automatically apply results to context

## 7.14 hifa\_tsysflagspectra

Deviant Tsys measurements are detected by analysis of a compressed view of the Tsys results. This view is a list of 2D images with axes 'Time' and 'Antenna' and value determined by the metric parameter. There is one image for each spectral window and polarization and intent group in the set: ['ATMOSPHERE'] (this default grouping of intents can be overridden using the parameter 'intentgroups').

If metric = 'median' then each pixel in each image is the median of the Tsys spectrum obtained at that point. If metric = 'shape' then each pixel in each image is a measure of the difference in the shape of the Tsys spectrum there from the shape of the median of all the Tsys spectra contributing to the image. The shape metric is computed as:

shape = 100 \* mean(abs(normalized Tsys - median normalized Tsys))

and a normalized array is:

normalized = array / median (array)

If metric = 'derivative' then each pixel in each image is 100 \* the MAD (median absolute deviation) of the channel by channel derivative of the Tsys spectrum.

If metric = 'fieldshape' then each pixel in each image is a measure of the difference in the shape of the Tsys spectrum there from the shape of the median of all Tsys spectra for that antenna with refintent.

The shape metric is computed as:

shape = 100 \* mean(abs(normalized Tsys - reference normalized Tsys))

and a normalized array is:

normalized = array / median (array)

Flags are generated by running the following rules on each image:

If flag\_nmedian = True then pixels greater than fnm\_limit \* median of all the pixels in the image are flagged.

If flag\_hi = True then pixels are flagged if they are greater

than  $\text{median} + \text{fhi\_limit} * \text{MAD}$  where median and MAD are computed for all the pixels in the image. No flags are set if there are fewer than `fhi_minsample` pixels in the sample.

If `flag_maxabs` = True then pixels are flagged if they have an absolute value greater than `fmax_limit`.

If `flag_minabs` = True then pixels are flagged if they have an absolute value less than `fmin_limit`.

If `flag_tmf1` = True then antennas with too many flagged points as defined by `tmf1_limit` are flagged entirely.

## Task Description

Flag deviant system temperature measurements

`hif_tsystab` flags deviant system temperature measurements in the system temperature calibration table.

### Keyword arguments:

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically.

In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.  
default: 'automatic'.

### ---- pipeline parameters that can be set in any pipelinemode

**flag\_nmedian** -- True to flag figures of merit greater than `fnm_limit * median`.  
default True

**fnm\_limit** -- Points greater than `fnm_limit * median` of samples are flagged.  
default 2

**flag\_hi** -- True to flag high figure of merit outliers.  
default: False

**fhi\_limit** -- Sample points beyond `fhi_limit * median` absolute deviation (MAD) of the figure of merit are flagged.  
default: 5.0

**fhi\_minsample** -- If there are too few points in the sample the statistics are assumed to be unreliable and no flagging is done.  
default: 5

**flag\_maxabs** -- True to flag figures of merit with absolute value above `fmax_limit`.  
default: False

**fmax\_limit** -- Figures of merit with absolute value above `fmax_limit` are flagged.  
default: 5.0

**flag\_minabs** -- True to flag figures of merit with absolute value below `fmin_limit`.  
default: False

**fmin\_limit** -- Figures of merit with absolute value below `fmin_limit` are flagged.  
default: 5.0

**flag\_tmf1** -- True to flag antennas that have too many flagged points as a function of time.  
default: False

**tmf1\_axis** -- Name of axis being flagged (Cannot be changed at present).  
default: Time

**tmf1\_limit** -- Fraction of flagged antennas points that triggers flagging the entire antenna.  
default: 0.5

---- pipeline-defined parameters which can be set only in 'interactive mode'

**caltable** -- List of input Tsys calibration tables  
default: [] - Use the table currently stored in the pipeline context.  
example: caltable=['X132.ms.tsys.s2.tbl']

**metric** -- Method to use as a measure of the Tsys quality. Possible values are:  
'median' - Each pixel in each flagging image is the median value of the unflagged channels in the associated Tsys spectrum.  
'shape' - Each pixel in each flagging image is a measure of the difference in shape of the unflagged part of the associated normalised Tsys spectrum from the median of the normalised Tsys spectra contributing to the image. The shape metric is computed as:  
$$\text{shape} = 100 * \text{mean}(\text{abs}(\text{normalized Tsys} - \text{median normalized Tsys}))$$
where a normalized array is:  
$$\text{normalized} = \text{array} / \text{median}(\text{array})$$
  
'fieldshape' - Each pixel in each flagging image is a measure of the difference in shape of the unflagged part of the associated normalised Tsys spectrum from the median of the normalised Tsys spectra for that antenna for refintent. The shape metric is computed as:  
$$\text{shape} = 100 * \text{mean}(\text{abs}(\text{normalized Tsys} - \text{reference normalized Tsys}))$$
where a normalized array is:  
$$\text{normalized} = \text{array} / \text{median}(\text{array})$$
  
'derivative' - Each pixel in each flagging image is  $100 * \text{the MAD}$  (median absolute deviation) of the channel by channel derivative of the associated normalised Tsys spectrum.  
default: 'median'

**intentgroups** -- How the data from various intents are to be combined in the flagging views. Internally the intentgroups are stored in a list of strings but, because of interface limitations, this list must be input as a single string.  
default: "[ATMOSPHERE]"  
For each spw/pol the default creates a separate 2d (ANTENNA v TIME) array of the Tsys metric for all data with ATMOSPHERE intent.  
example: "[AMPLITUDE + BANDPASS + PHASE + TARGET]"  
For each spw/pol this would create a 2d array of the Tsys metric using data for all the listed intents combined - useful if you are looking for changes that depend on intent.

**refintent** -- When metric='fieldshape' this specifies the data intent whose Tsys provide the 'reference' shape for comparison.  
default: 'BANDPASS'  
example: "[AMPLITUDE + BANDPASS + PHASE + TARGET]"  
The median of data with all these intents will be used as the 'reference'.

## -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

## Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

## Examples

1. Flag Tsys measurements where the Tsys median is greater than 2 \* the median of this metric over the flagging image.

This is currently the default.

hif\_tsysflagspectra()

2. Flag Tsys measurements whose 'shape' metric lies more than 5 \* MAD above the median of this metric over the flagging image.

hif\_tsysflagspectra(metric='shape', flag\_hi=True, fhi\_limit=5, flag\_nmedian=False)

3. Flag Tsys measurements whose 'derivative' metric is greater than 2.

hif\_tsysflagspectra(metric='derivative', flag\_maxabs=True, fmax\_limit=2, flag\_nmedian=False)

## Parameter List

Table 25: hifa\_tsysflagspectra default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets (Not used)
<b>citable</b>	stringArray	None	List of input ctables
<b>metric</b>	string	median	Method used to judge the quality of a Tsys measurement
<b>intentgroups</b>	string	"ATMOSPHERE"	List of groups of intents whose Tsys are to be compared with the reference
<b>refintent</b>	string	BANDPASS	Data intent that provides the reference shape for 'fieldshape' metric
<b>flag_hi</b>	bool	False	True to flag high figure of merit outliers
<b>fhi_limit</b>	double	5.0	Flag figure of merit values higher than fhi_limit *MAD
<b>fhi_minsample</b>	int	5	Minimum number of samples for valid MAD estimate
<b>flag_maxabs</b>	bool	False	True to flag pixels with absolute

			value higher than fmax_limit
<b>fmax_limit</b>	double	5.0	Flag pixels whose absolute value is higher than this limit
<b>flag_minabs</b>	bool	False	True to flag pixels with absolute value lower than fmin_limit
<b>fmin_limit</b>	double	5.0	Flag pixels whose absolute value is lower than this limit
<b>flag_nmedian</b>	bool	True	True to flag figure of merit values higher than fnm_limit * median
<b>fnm_limit</b>	double	2.0	Flag figure of merit values higher than fnm_limit *median
<b>flag_tmf1</b>	bool	False	True to flag data with too many flagged samples
<b>tmf1_axis</b>	string	Time	Name of flagging samples axis
<b>tmf1_limit</b>	double	0.5	Proportion of samples flagged that triggers too many flags rule
<b>niter</b>	int	1	Number of view calculation/flag iterations
<b>pipelinemode</b>	string	automatic	The pipeline operations mode
<b>dryrun</b>	bool	False	Run the task (False) or list commands(True)
<b>acceptresults</b>	bool	True	Automatically apply results to context

## 7.15 hifa\_tsysflag

### Task Description

Flag deviant system temperature measurements

hif\_tsysflag tries to flag all deviant system temperature measurements in the system temperature calibration table. It does this by running a sequence of flagging tests, each designed to look for a different type of error. The tests are:

1. Flag Tsys spectra with high median values by running sister task hifa\_tsysflagspectra with metric='median', flag\_nmedian=True, fnm\_limit=fnm\_limit.
2. Flag Tsys spectra with high median derivatives by running hifa\_tsysflagspectra with metric='derivative', flag\_maxabs=True, fmax\_limit=fd\_max\_limit. This is meant to spot spectra that are 'ringing'.
3. Flag the edge channels of the Tsys spectra in each SpW by running hifa\_tsysflagchans with intentgroupss=['ATMOSPHERE','BANDPASS','AMPLITUDE'], flag\_edges=True, edge\_limit=fe\_edge\_limit.
4. Flag Tsys spectra whose shape is different from that associated with the BANDPASS intent or which are associated with an antenna that has been too heavily flagged already, by running hif\_tsysflagspectra with metric='fieldshape', refintent=ff\_refintent, flag\_maxabs=True, fmax\_limit=ff\_max\_limit, flag\_tmf1=True, tmf1\_axis='Antenna1', tmf1\_limit=ff\_tmf1\_limit.
5. Flag 'birdies' by running hif\_tsysflagchans with metric='antenna\_diff', flag\_sharps=True, sharps\_limit=fb\_sharps\_limit.

## **Keyword arguments:**

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline determines the values of all context defined pipeline inputs automatically.

In interactive mode the user can set the pipeline-defined parameters manually. In 'getinputs' mode the user can check the settings of all pipeline parameters without running the task.

default: 'automatic'.

### **---- pipeline-defined parameters which can be set only in 'interactive mode'**

**caltable** -- List of input Tsys calibration tables

default: [] - Use the table currently stored in the pipeline context.

example: caltable=['X132.ms.tsys.s2.tbl']

**flag\_nmedian** -- True to flag Tsys spectra with high median value.

default: True

**fnm\_limit** -- Flag spectra with median value higher than fnm\_limit \* median of this measure over all spectra.

default: 2.0

**flag\_derivative** -- True to flag Tsys spectra with high median derivative.

default: True

**fd\_max\_limit** -- Flag spectra with median derivative higher than fd\_max\_limit \* median of this measure over all spectra.

default: 5.0

**flag\_edgechans** -- True to flag edges of Tsys spectra.

default: True

**fe\_edge\_limit** -- Flag channels whose channel to channel difference > fe\_edge\_limit \* median across spectrum.

default: 3.0

**flag\_fieldshape** -- True to flag Tsys spectra with a radically different shape to those of the ff\_refintent.

default: True

**ff\_refintent** -- Data intent that provides the reference shape for 'flag\_fieldshape'.

default: BANDPASS

**ff\_max\_limit** -- Flag Tsys spectra with 'fieldshape' metric values > ff\_max\_limit.

default: 5.0

**ff\_tmf1\_limit** -- Flag all Tsys spectra for an antenna if proportion flagged already > ff\_tmf1\_limit.

default: 0.666

**flag\_birdies** -- True to flag channels covering sharp spectral features.

default: True

**fb\_sharps\_limit** -- Flag channels bracketing a channel to channel difference > fb\_sharps\_limit.

default: 0.05

## -- Pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: True

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

## Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

## Examples

1. Flag Tsys measurements using currently recommended tests:

hif\_tsysflag()

2. Flag Tsys measurements using all recommended tests apart from that using the 'fieldshape' metric.

hif\_tsysflag(flag\_fieldshape=False)

## Parameter List

Table 26: hifa\_tsysflag default settings

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input measurement sets (Not used)
<b>citable</b>	stringArray	None	List of input ctables
<b>flag_nmedian</b>	bool	True	True to flag Tsys spectra with high median value
<b>fnm_limit</b>	double	2.0	Flag spectra with median > fnm_limit * median over all spectra
<b>flag_derivative</b>	bool	True	True to flag Tsys spectra with high median derivative
<b>fd_max_limit</b>	double	5.0	Flag spectra with median derivative higher than fd_max_limit * median of this measure over all spectra
<b>flag_edgechans</b>	bool	True	True to flag edges of Tsys spectra
<b>fe_edge_limit</b>	double	3.0	Flag channels whose channel to channel difference >

			fe_edge_limit * median across spectrum
<b>flag_fieldshape</b>	bool	True	True to flag Tsys spectra with a radically different shape to those of the ff_refintent
<b>ff_refintent</b>	string	BANDPASS	Data intent providing the reference shape for '\flag_fieldshape\'
<b>ff_max_limit</b>	double	5.0	Flag Tsys spectra with '\fieldshape' metric > ff_max_limit
<b>ff_tmf1_limit</b>	double	0.666	Flag all Tsys spectra for an antenna if proportion flagged already ff_tmf1_limit
<b>flag_birdies</b>	bool	True	True to flag channels covering sharp spectral features
<b>fb_sharps_limit</b>	double	0.05	Flag channels bracketing a channel to channel difference > fb_sharps_limit
<b>pipelinemode</b>	string	automatic	The pipeline operations mode
<b>dryrun</b>	bool	False	Run the task (False) or list commands (True)
<b>acceptresults</b>	bool	True	Automatically apply results to context

## 7.16 *hifa\_wvrgcalflag*

### Task Description

First, generate a gain table based on the Water Vapour Radiometer data in each vis file. Second, apply the wvr calibration to the data specified by 'flag\_intent', calculate flagging 'views' showing the ratio phase-rms with wvr / phase-rms without wvr for each scan. A ratio < 1 implies that the phase noise is improved, a score > 1 implies that it is made worse. Third, search the flagging views for antennas with anomalous high values. If any are found then recalculate the wvr calibration with the 'wvrflag' parameter set to ignore their data and interpolate results from other antennas according to 'maxdistrm' and 'minnumants'. Fourth, if the overall QA score for the final wvr correction of a vis file is greater than the value in 'accept\_threshold' then make available the wvr calibration file for merging into the context and use in the subsequent reduction.

**vis** -- List of input visibility files

default: none, in which case the vis files to be used will be read from the context.

example: vis=['ngc5921.ms']

**caltable** -- List of output gain calibration tables

default: none, in which case the names of the caltables will be generated automatically.

example: caltable='ngc5921.wvr'

**hm\_toffset** -- If 'manual', set the 'toffset' parameter to the user-specified value.

If 'automatic', set the 'toffset' parameter according to the date of the measurement set; toffset=-1 if before 2013-01-21T00:00:00 toffset=0 otherwise.

default: 'automatic'

toffset -- Time offset (sec) between interferometric and WVR data

default: 0

**segsource** -- If True calculate new atmospheric phase correction coefficients for each source, subject to the constraints of the 'tie' parameter. 'segsource' is forced to be True if the 'tie' parameter is set to a non-empty value by the user or by the automatic heuristic.

default: True

**hm\_tie** -- If 'manual', set the 'tie' parameter to the user-specified value.

If 'automatic', set the 'tie' parameter to include with the target all calibrators that are within 15 degrees of it:  
if no calibrators are that close then 'tie' is left empty.

default: 'automatic'

**tie** -- Use the same atmospheric phase correction coefficients when calculating the wvr correction for all sources in the 'tie'. If 'tie' is not empty then 'segsource' is forced to be True. Ignored unless hm\_tie='manual'.

default: []

example: ['3C273,NGC253', 'IC433,3C279']

**sourceflag** -- Flag the WVR data for these source(s) as bad and do not produce corrections for it. Requires segsource=True

default: []

example: ['3C273']

**nsol** -- Number of solutions for phase correction coefficients during this observation, evenly distributed in time throughout the observation. It is used only if segsource=False because if segsource=True then the coefficients are recomputed whenever the telescope moves to a new source (within the limits imposed by 'tie').

default: 1

**disperse** -- Apply correction for dispersion

default: False

**wvrfalg** -- Flag the WVR data for these antenna(s) as bad and replace its data with interpolated values

default: []

example: ['DV03','DA05','PM02']

**hm\_smooth** -- If 'manual' set the 'smooth' parameter to the user-specified value.

If 'automatic', run the wvrgcal task with the range of 'smooth' parameters required to match the integration time of the wvr data to that of the

interferometric data in each spectral window.

**smooth** -- Smooth WVR data on this timescale before calculating the correction.  
Ignored unless hm\_smooth='manual'.

default: '1s'

**scale** -- Scale the entire phase correction by this factor.

default: 1

**maxdistm** -- Maximum distance in meters of an antenna used for interpolation from a flagged antenna.

default: 500

example: 550

**minnumants** -- Minimum number of nearby antennas (up to 3) used for interpolation from a flagged antenna.

default: 2

example: 3

**flag\_intent** -- The data intent(s) on whose wvr correction results the search for bad wvr antennas is to be based.

A 'flagging view' will be calculated for each specified intent, in each spectral window in each vis file.

Each 'flagging view' will consist of a 2-d image with dimensions ['ANTENNA', 'TIME'], showing the phase noise after the wvr correction has been applied.

If flag\_intent is left blank, the default, the flagging views will be derived from data with the default bandpass calibration intent i.e. the first in the list BANDPASS, PHASE, AMPLITUDE for which the measurement set has data.

default "

**qa\_intent** -- The list of data intents on which the wvr correction is to be tried as a means of estimating its effectiveness.

A QA 'view' will be calculated for each specified intent, in each spectral window in each vis file.

Each QA 'view' will consist of a pair of 2-d images with dimensions ['ANTENNA', 'TIME'], one showing the data phase-noise before the wvr application, the second showing the phase noise after (both 'before' and 'after' images have a bandpass calibration applied as well).

An overall QA score is calculated for each vis file, by dividing the 'before' images by the 'after' and taking the median of the result. An overall score of 1 would correspond to no change in the phase noise, a score > 1 implies an improvement.

If the overall score for a vis file is less than the value in 'accept\_threshold' then the wvr calibration file is not made available for merging into the context for use in the subsequent reduction.

default: 'BANDPASS,PHASE'

**qa\_bandpass\_intent** -- The data intent to use for the bandpass calibration in the qa calculation. The default is blank to allow the underlying bandpass task to select a sensible intent if the dataset lacks BANDPASS data.

default: "

**accept\_threshold** -- The phase-rms improvement ratio (rms without wvr / rms with wvr) above which the wrvg file will be accepted into the context for subsequent application.

default: 1.0

## Examples

1. Compute the WVR calibration for all the measurement sets.  
hifa\_wvrgcalflag (hm\_tie='automatic')

## Parameter List

**Table 27: hifa\_wvrgcalflag default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input visibility files
<b>citable</b>	stringArray	None	List of output gain calibration tables
<b>hm_toffset</b>	string	automatic	Toffset computation heuristic method
<b>toffset</b>	double	0	Time offset (sec) between IF and WVR data
<b>segsource</b>	bool	True	Compute new coefficient calculation for each source
<b>sourceflag</b>	stringArray	None	Flag the WVR data for these source(s)
<b>hm_tie</b>	string	automatic	Tie computation heuristics method
<b>tie</b>	stringArray	None	Sources for which to use the same atmospheric phase correction coefficients
<b>nsol</b>	int	1	Number of solutions for phase correction coefficients
<b>disperse</b>	bool	False	Apply correction for dispersion
<b>wvrflag</b>	stringArray	None	Flag the WVR data for these antenna(s) replace with interpolated values
<b>hm_smooth</b>	string	automatic	Smoothing computation heuristics method
<b>smooth</b>	string	1s	Smooth WVR data on the given timescale before calculating the correction
<b>scale</b>	double	1.	Scale the entire phase correction by this factor
<b>maxdistrm</b>	double	500.	Maximum distance (m) of an antenna used for interpolation for a flagged antenna

<b>minnumants</b>	int	2	Minimum number of near antennas (up to 3) required for interpolation
<b>mingoodfrac</b>	double	0.8	Minimum fraction of good data per antenna antenna
<b>flag_intent</b>	string	None	Data intents to use in detecting and flagging bad wvr antennas
<b>qa_intent</b>	string	BANDPASS,PHASE	Data intents to use in estimating the effectiveness of the wvr correction
<b>qa_bandpass_intent</b>	string	None	Data intent to use for the bandpass calibration in the qa calculation
<b>accept_threshold</b>	double	1.0	Improvement ratio (phase-rms without wvr / phase-rms with wvr) above which wvrg calibration file will be accepted
<b>flag_hi</b>	bool	True	True to flag high figure of merit outliers
<b>fhi_limit</b>	double	10.0	Flag figure of merit values higher than limit *MAD
<b>fhi_minsample</b>	int	5	Minimum number of samples for valid MAD estimate
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## 7.17 hifa\_wvrgcal

### Task Description

Generate a gain table based on the Water Vapour Radiometer data in each vis file. By applying the wvr calibration to the data specified by 'qa\_intent' and 'qa\_spw', calculate a QA score to indicate its effect on interferometric data; a score > 1 implies that the phase noise is improved, a score < 1 implies that it is made worse. If the score is less than 'accept\_threshold' then the wvr gain table is not accepted into the context for subsequent use.

**vis** -- List of input visibility files

default: none, in which case the vis files to be used will be read from the context.

example: vis=['ngc5921.ms']  
caltable -- List of output gain calibration tables  
default: none, in which case the names of the caltables will be generated automatically.  
example: caltable='ngc5921.wvr'

**hm\_toffset** -- If 'manual', set the 'toffset' parameter to the user-specified value. If 'automatic', set the 'toffset' parameter according to the date of the measurement set; toffset=-1 if before 2013-01-21T00:00:00 toffset=0 otherwise.  
default: 'automatic'

**toffset** -- Time offset (sec) between interferometric and WVR data  
default: 0

**segsource** -- If True calculate new atmospheric phase correction coefficients for each source, subject to the constraints of the 'tie' parameter. 'segsource' is forced to be True if the 'tie' parameter is set to a non-empty value by the user or by the automatic heuristic.  
default: True

**hm\_tie** -- If 'manual', set the 'tie' parameter to the user-specified value. If 'automatic', set the 'tie' parameter to include with the target all calibrators that are within 15 degrees of it:  
if no calibrators are that close then 'tie' is left empty.  
default: 'automatic'

**tie** -- Use the same atmospheric phase correction coefficients when calculating the wvr correction for all sources in the 'tie'. If 'tie' is not empty then 'segsource' is forced to be True. Ignored unless hm\_tie='manual'.  
default: []  
example: ['3C273,NGC253', 'IC433,3C279']

**sourceflag** -- Flag the WVR data for these source(s) as bad and do not produce corrections for it. Requires segsource=True.  
default: []  
example: ['3C273']

**nsol** -- Number of solutions for phase correction coefficients during this observation, evenly distributed in time throughout the observation. It is used only if segsource=False because if segsource=True then the coefficients are recomputed whenever the telescope moves to a new source (within the limits imposed by 'tie').  
default: 1

**disperse** -- Apply correction for dispersion  
default: False

**wvrfalg** -- Flag the WVR data for the listed antennas as bad and replace their data with values interpolated from the 3 nearest antennas with unflagged data.  
default: []  
example: ['DV03','DA05','PM02']

**hm\_smooth** -- If 'manual' set the 'smooth' parameter to the user-specified value. If 'automatic', run the wvrgcal task with the range of 'smooth' parameters required to match the integration time of the wvr data to that of the interferometric data in each spectral window.

smooth -- Smooth WVR data on this timescale before calculating the correction. Ignored unless hm\_smooth='manual'.  
default: '1s'

**scale** -- Scale the entire phase correction by this factor.

default: 1

maxdistrm -- Maximum distance in meters of an antenna used for interpolation from a flagged antenna.

default: 500

example: 550

**minnumants** -- Minimum number of nearby antennas (up to 3) used for interpolation from a flagged antenna.

default: 2

example: 3

**qa\_intent** -- The list of data intents on which the wvr correction is to be tried as a means of estimating its effectiveness.

A QA 'view' will be calculated for each specified intent, in each spectral window in each vis file.

Each QA 'view' will consist of a pair of 2-d images with dimensions ['ANTENNA', 'TIME'], one showing the data phase-noise before the wvr application, the second showing the phase noise after (both 'before' and 'after' images have a bandpass calibration applied as well).

An overall QA score is calculated for each vis file, by dividing the 'before' images by the 'after' and taking the median of the result. An overall score of 1 would correspond to no change in the phase noise, a score > 1 implies an improvement.

If the overall score for a vis file is less than the value in 'accept\_threshold' then the wvr calibration file is not made available for merging into the context for use in the subsequent reduction.

If you do not want any QA calculations then set qa\_intent="".

default: "

example: 'PHASE'

**qa\_bandpass\_intent** -- The data intent to use for the bandpass calibration in the qa calculation. The default is blank to allow the underlying bandpass task to select a sensible intent if the dataset lacks BANDPASS data.

default: "

**qa\_spw** -- The SpW(s) to use for the qa calculation, in the order that they should be tried. Input as a comma-separated list.

The default is blank, in which case the task will try SpWs in order of decreasing median sky opacity.

default: "

**accept\_threshold** -- The phase-rms improvement ratio

(rms without wvr / rms with wvr) above which the wrvg file will be accepted into the context for subsequent application.

default: 1.0

## Examples

1. Compute the WVR calibration for all the measurement sets.  
**hifa\_wvrgcal (hm\_tie='automatic')**

## Parameter List

**Table 45: hifa\_wvrgcal default settings**

Parameter	Type	Default	Description
<b>vis</b>	stringArray	None	List of input visibility files
<b>caltable</b>	stringArray	None	List of output gain calibration tables
<b>hm_toffset</b>	string	automatic	Toffset computation heuristic method
<b>toffset</b>	double	0	Time offset (sec) between IF and WVR data
<b>segsource</b>	bool	True	Compute new coefficient calculation for each source
<b>sourceflag</b>	stringArray	None	Flag the WVR data for these source(s)
<b>hm_tie</b>	string	automatic	Tie computation heuristics method
<b>tie</b>	stringArray	None	Sources for which to use the same atmospheric phase correction coefficients
<b>nsol</b>	int	1	Number of solutions for phase correction coefficients
<b>disperse</b>	bool	False	Apply correction for dispersion
<b>wvrflag</b>	stringArray	None	Flag the WVR data for these antenna(s) replace with interpolated values
<b>hm_smooth</b>	string	automatic	Smoothing computation heuristics method
<b>smooth</b>	string	1s	Smooth WVR data on the given timescale before calculating the correction
<b>scale</b>	double	1.	Scale the entire phase correction by this factor
<b>maxdistrm</b>	double	500.	Maximum distance (m) of an antenna used for interpolation for a flagged antenna
<b>minnumants</b>	int	2	Minimum number of near antennas (up to

			3) required for interpolation
<b>mingoodfrac</b>	double	0.8	Minimum fraction of good data per antenna
<b>qa_intent</b>	string	None	Data intents to use in estimating the effectiveness of the wvr correction
<b>qa_bandpass_intent</b>	string	None	Data intent to use for the bandpass calibration in the qa calculation
<b>qa_spw</b>	string	None	Data SpW(s) to use in estimating the effectiveness of the wvr correction
<b>accept_threshold</b>	double	1.0	Improvement ratio (phase-rms without wvr / phase-rms with wvr) above which wvrg calibration file will be accepted
<b>pipelinemode</b>	string	automatic	The pipeline operating mode
<b>dryrun</b>	bool	False	Run the task (False) or display the command (True)
<b>acceptresults</b>	bool	True	Add the results to the pipeline context

## 8 Single-Dish Task Descriptions

### 8.1 *hsd\_applycal*

*hsd\_applycal* is used to apply a list of Tsys calibration tables generated by *hifa\_tsycal* and atmospheric calibration tables, i.e. OFF position spectra generated by *hsd\_calsky*. This task employed the task *sdcal2*, wherein (ON - OFF)/ OFF \* Tsys is calculated at once. Here “ON” and “OFF” are spectra on ON- and OFF-source positions, respectively, “Tsys” is an averaged Tsys value over frequency range of a science spectral window (excluding flagged channels).

#### Task Description

Apply calibration tables to the data for science target data

#### ---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**infiles** -- List of data files. The data file names must be those of Scantables (i.e. files with \*.asap suffix) that are registered to context via hsd\_mstoscantable task.

default: [] (=all data in context)

example: infiles=['X227.PM01.asap', 'X227.PM02.asap']

**field** – The list of field name(s). Defaults to all files in context.

default: " " (=all fields)

**spw** – The list of spectral window(s).

default: " " (=all spectral windows)

**scan** – The list of scan number(s).

default: " " (=all scans)

**pol** – The list of polarization(s).

default: " " (=all polarizations)

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

#### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

#### Examples

1. Generate caltables for all data managed by context.

```
hsd_applycal(pipelinemode='automatic')
```

#### Parameter List

**Table 46: hsd\_applycal default settings**

Parameter	Type	Default	Description
<b>infiles</b>	StringArray	None	List of input files to be calibrated
<b>field</b>	String	None	Field to be calibrated
<b>spw</b>	String	None	select data by IF IDs

<b>scan</b>	String	None	select data by scan numbers
<b>pol</b>	String	None	select data by polarizations
<b>dryrun</b>	Bool	False	Run the task (False) or display task (True)
<b>acceptresults</b>	Bool	True	Add the results into the pipeline context

## 8.2 *hsd\_baseline*

WARNING: *hsd\_baseline* should be executed on science (line) data, not on amplitude calibrator (continuum) data.

*hsd\_baseline* is used to generate baseline fitting tables and to subtract baselines from the spectra. This task contains an automated line identification algorithm using clustering analysis. The automated line identification algorithm searches for candidate emission or absorption lines, and masks them. Then Pipeline performs baseline fitting using a cubic spline that connects an empirically defined number of segments of each spectrum.

Line identification is performed on the integrated data of all EBs (per spectral window). As shown in Fig. 1, the *hsd\_baseline* task first distributes all spectra into three groups in the time-domain, and splits the spectra per each group into XY-grids (grid cell size is equivalent beam size). That is, all spectra will be distributed into each voxel per time-domain group and per XY-grid. Then the Pipeline searches for emission line candidates having a brightness above a certain threshold (based on the median absolute deviation) on the integrated spectrum per voxel. Line detection validation is performed by checking whether how many voxels have line candidates along the time-axis, e.g., if a XY-grid cell is detected in the three time-domain groups (in the three voxels along the time-axis), then the grid cell is considered to contain emission (marked as “True”, shown in blue on the right panels of Fig.1). If a XY-grid cell is detected in two time-domain groups and not detected in another time-domain group, it is considered “Probable” (possibly detected, shown in green on the right panels of Fig.1).

Then clustering analysis is performed on the central frequency vs. line-width plane to determine the number of emission lines and the cluster member. In CASA 4.5.1 Pipeline, the number of emission lines is determined using k-means clustering. Finally, emission line channels of each spectrum are defined by taking into account the spatial distribution of the line center and the line width in the XY-grids. The line center and the line width for each grid cell are assigned to the values for the spectrum at the center of the grid cell. For other grid cell where emission line was not detected in the algorithm, the line center and linewidth are calculated by interpolating from those of the pointing direction of the spectrum.

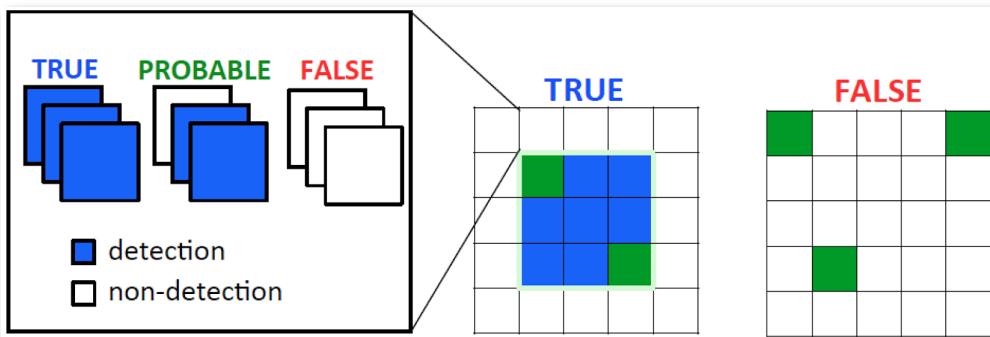
In CASA4.5.1, another masking algorithm is also employed in addition to the line finding algorithm ('deviation mask'). This algorithm is to deter the fitting baseline from being dragged by unwanted features such as atmospheric absorption lines and lines at the band edge. The channel masks are created based on the Standard Deviation (STD) calculated for each channels. The process detects strong features like astronomical lines, atmospheric lines, and band edges. The process is done for subgroup of spectra splitted by EB, Antenna, and spectral

window. The algorithm performs the following steps in order:

1. Subtract median value from each spectrum to reduce offset
2. Calculate STD for each channel along time series to create a STD spectrum: STD[channel]
3. Determine two thresholds (detection\_threshold and extention\_threshold) based on the STD of STD[channel] above.
  - detection\_threshold =  $5.0 \times \text{STD}(\text{STD[channel]})$
  - extention\_threshold =  $2.0 \times \text{STD}(\text{STD[channel]})$
4. Create masks for channels where  $\text{STD[channel]} > \text{detection\_threshold}$
5. For each detected channels (masks), masked area is extended toward both sides until  $\text{STD[channel]}$  gets lower than extention\_threshold.

Note that all spectra in the subgroup will have the same deviation masks, and the same deviation masks are used repeatedly even if baseline\_subtraction stage is performed multiple times (iteration case).

This task (and the following task, hsd\_bfflag) is executed twice in the standard procedure to improve accuracy of the line finding algorithm. The number of iteration is defined as twice by default because the data used for the line identification in the first iteration is the one without baseline subtraction, while the one used for the second iteration is the one after subtraction at the first hsd\_baseline. In almost all the case, the line finding algorithm is improved at second iteration. After determining masked ranges, the baseline subtraction is always done on the calibrated data without using previously-obtained baseline or baseline flag information. In other words, even at the second iteration, both the baseline and flag information obtained at the first iteration are all cleared out, then the fitted baseline is subtracted again from the originally calibrated data using the baseline and flag information obtained at second iteration.



**Fig. 1** Line detection validation scheme on the spatial plane in task hsd\_baseline. The spectrum belonging to each grid cell is split into three time-domain groups (voxel).

### Task Description

Detect and validate spectral lines, subtract baseliens by masking detected lines

---- pipeline-defined parameters which can be set in any pipelinemode

**fitfunc** -- fitting function for baseline subtraction. You can choose cubic spline ('spline' or 'cspline')  
default: 'cspline'.

**fitorder** -- Fitting order for polynomial. For cubic spline, this is used to determine the number of segments the spectrum is divided into. Default (-1) is to determine the order automatically.  
default: -1 (auto determination)

**linewindow** -- Pre-defined line window. If this is set, specified line windows are used as a emission line to be masked for baseline subtraction (i.e., line detection and validation stage will be skipped). Two types of format are acceptable: (1) channel-based format, [min\_chan,max\_chan] and (2) a format with combination of central frequency (in GHz) and velocity range (in km/s), [center\_freq, min\_vel, max\_vel]. You can set multiple line windows by using nested list. However, the two types of format must not be mixed in the list for multiple line windows.

default: [] (do line detection and validation)

example: [100.0,200.0], [115.0,-10.0,10.0]

**edge** -- number of edge channels to be ignored at baseline subtraction. The parameters must be given in a list of 2 integers, denoting the numbers of left- and right- edge channels, respectively.

default: [] ([0,0])

example: [10,10]

**broadline** -- Try to detect broad component of spectral line if True.

default: True

**clusteringalgorithm** -- Selection of the algorithm in the clustering analysis to use for the validation of detected lines. 'kmean' algorithm and hierarchical clustering algorithm 'hierarchy' are so far implemented.

default: 'kmean'.

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.

default: 'automatic'.

#### ---- pipeline-defined parameter which can be set only in 'interactive mode'

**infiles** -- List of data files. The data file names must be those of Scantables (i.e. files with \*.asap suffix) that are registered to context via hsd\_mstoscantable task.

default: [] (all data in context)

example: infiles=['X227.PM01.asap', 'X227.PM02.asap']

**spw** -- Select spectral window.

default: " (all spectral windows)

**pol** -- Select polarization.

default: " (all polarizations)

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: Flase

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Examples

```
hsd_baseline(pipelinemode='automatic')
```

### Parameter List

**Table 47: hsd\_baseline default settings**

Parameter	Type	Default	Description
<b>fitfunc</b>	String	'cspline'	Fitting function for baseline subtraction
<b>fitorder</b>	String	None	Fitting order for baseline subtraction
<b>linewindow</b>	String	None	Pre-defined line window
<b>edge</b>	String	None	Edge channels to be dropped
<b>broadline</b>	Bool	True	Try to detect broad component of the line
<b>clusteringalgorithm</b>	String	'kmean'	Algorithm used in clustering analysis
<b>infiles</b>	StringArray	None	List of input files to be calibrated
<b>spw</b>	String	None	Set of data selection by spectral windows
<b>pol</b>	String	None	Set of data selection by polarizations
<b>dryrun</b>	Bool	False	Run the task (False) or display task (True)
<b>acceptresults</b>	Bool	True	Add the results into the pipeline context

### 8.3 hsd\_bfflag

WARNING: hsd\_bfflag should be executed on science (line) data, not on amplitude calibrator (continuum) data.

hsd\_bfflag is used to examine spectra before and after baseline subtraction and to flag out the deviators based on Tsys and baseline RMS. All of the threshold used in this task is determined empirically, and can be changed manually. The flagging criteria is as follows:

[1] Abnormally high Tsys (per spectrum) :

Tsys per spectrum is averaged over science frequency range. The data with Tsys larger than 3 times standard deviation of Tsys per EB per antenna and polarization per spectral window, shall be flagged.

e.g. The Tsys of the data to be flagged is expressed by

$$Tsys(n) > 3\sigma_{Tsys} + \overline{Tsys},$$

where  $Tsys(n)$  is the Tsys assigned to the n-th spectrum,  $\sigma_{Tsys} (= \sqrt{\frac{\sum_n (Tsys(n) - \overline{Tsys})^2}{N}})$  is the standard deviation of Tsys,  $\overline{Tsys} (= \frac{\sum_n Tsys(n)}{N})$  is the mean Tsys per EB per antenna and polarization per spectral window and  $N$  is total number of spectra per EB per antenna and polarization per spectral window.

[2] Large standard deviation of baseline RMS:

The baseline RMS of a single spectrum is calculated using emission-free channels (excluding masked channels of emission line).

e.g. The baseline RMS of the data to be flagged is expressed by

$$B_{rms}(n) > 4.5 \sigma_{Baseline} + \overline{B_{rms}} \text{ (before baseline subtraction), and}$$

$$B_{rms}(n) > 4.0 \sigma_{Baseline} + \overline{B_{rms}} \text{ (after baseline subtraction),}$$

where  $B_{rms}(n)$  is baseline RMS of n-th spectrum,  $\sigma_{Baseline}$  is the standard deviation of baseline RMS, and  $\overline{B_{rms}}$  is the averaged baseline RMS per EB per antenna and polarization per spectral window.

[3] Large deviation from the moving average:

The moving average (also known as running means) is computed with a window of 10 spectra. e.g. The moving average baseline RMS is defined as:

$$MAB_{rms}(n) = \frac{\sum_{n-5}^{n-1} B_{rms}(n) + \sum_{n+1}^{n+5} B_{rms}(n)}{10}.$$

The sample of deviation from the moving average is defined as

$$dB_{rms}(n) = B_{rms}(n) - MAB_{rms}(n).$$

The mean and standard deviation is expressed by

$$\overline{dB_{rms}} = \frac{\sum_n^N dB_{rms}(n)}{N} \text{ and}$$

$$\sigma_{MA} = \sqrt{\frac{\sum_n (dB_{rms}(n) - \overline{dB_{rms}})^2}{N}}, \text{ respectively.}$$

The baseline RMS to be flagged out is expressed by

$$dB_{rms}(n) > 5.5 \sigma_{MA} + \overline{dB_{rms}} \text{ (before baseline subtraction), and}$$

$$dB_{rms}(n) > 5.0 \sigma_{MA} + \overline{dB_{rms}} \text{ (after baseline subtraction).}$$

[4] Larger than expected RMS:

The expected RMS is defined as

$$RMS = \frac{Tsys}{\sqrt{dv*tinit}},$$

where Tsys is observed Tsys, dv is effective frequency resolution, and  $t_{init}$  is integration time per spectrum.)

### Task Description

Flag out outliers based on predefined criteria of single-dish pipeline

#### ---- pipeline parameters that can be set in any pipelinemode

**iteration** -- Number of iterations to perform sigma clipping to calculate threshold value of flagging.

default: 5

**edge** -- Number of channels to be dropped from the edge. The value must be a list of integer with length of one or two. If list length is one, same number will be applied both side of the band.

default: [0,0]

example: [10,20], [10]

**flag\_tsys** -- Activate (True) or deactivate (False) Tsys-based flag.

default: True

**tsys\_thresh** -- Threshold value for Tsys-based flag.

default: 3.0

**flag\_weath** -- Activate (True) or deactivate (False) weather-based flag. Since weather-based flag is not implemented yet, setting True has no effect at this moment.

default: False

**weath\_thresh** -- Threshold value for weather-based flag.

default: 3.0

**flag\_pfre** -- Activate (True) or deactivate (False) flagging based on expected RMS of pre-fit spectra.

default: True

**prfre\_thresh** -- Threshold value for flagging based on expected RMS of pre-fit spectra.  
default: 3.0

**flag\_pofre** -- Activate (True) or deactivate (False) flagging based on expected RMS of post-fit spectra.  
default: True

**pofre\_thresh** -- Threshold value for flagging based on expected RMS of post-fit spectra.  
default: 1.3333

**flag\_prfr** -- Activate (True) or deactivate (False) flagging based on RMS of pre-fit spectra.  
default: True

**prfr\_thresh** -- Threshold value for flagging based on RMS of pre-fit spectra.  
default: 4.5

**flag\_pofr** -- Activate (True) or deactivate (False) flagging based on RMS of post-fit spectra.  
default: True

**pofr\_thresh** -- Threshold value for flagging based on RMS of post-fit spectra.  
default: 4.0

**flag\_prfrm** -- Activate (True) or deactivate (False) flagging based on running mean of pre-fit spectra.  
default: True

**prfrm\_thresh** -- Threshold value for flagging based on running mean of pre-fit spectra.  
default: 5.5

**prfrm\_nmean** -- Number of channels for running mean of pre-fit spectra.  
default: 5

**flag\_pofrm** -- Activate (True) or deactivate (False) flagging based on running mean of post-fit spectra.  
default: True

**pofrm\_thresh** -- Threshold value for flagging based on running mean of post-fit spectra.  
default: 5.0

**pofrm\_nmean** -- Number of channels for running mean of post-fit spectra.  
default: 5

**flag\_user** -- Activate (True) or deactivate (False) user-defined flag. Since user flagging is not implemented yet, setting this True has no effect at the moment.  
default: False

**user\_thresh** -- Threshold value for flagging based on user-defined rule. Since user flagging is not implemented yet, setting this True has no effect at the moment.  
default: 3.0

**plotflag** – Set this True to plot the result of the data flagging.  
default: True

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode' or 'getinputs' modes

**infiles** -- List of data files. The data file names must be those of Scantables (i.e. files with \*.asap suffix) that are registered to context via hsd\_mstoscantable task.  
default: [] (process all data in context)

**spw** – The list of spectral window(s) to be processed. Defaults to all spectral windows.  
default: " (=all spectral windows)

**pol** – The list of polarization(s) to be processed. Defaults to all polarizations.  
default: " (=all polarizations)

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Parameter List

**Table 48: hsd\_bfiflag default settings**

Parameter	Type	Default	Description
<b>iteration</b>	int	5	Number of iteration to perform sigma clipping

			to calculate the threshold
<b>edge</b>	IntArray	None	Number of edge channels to be excluded from statistic calculation to flag the data
<b>flag_tsys</b>	Bool	True	Flag data based on Tsys
<b>tsys_thresh</b>	Double	3.0	Threshold for Tsys flag
<b>flag_pfre</b>	Bool	True	Flag data based on expected RMS of pre-fit spectra
<b>prfre_thresh</b>	Double	3.0	Threshold based on expected RMS of post-fit spectra flag
<b>flag_pofre</b>	Bool	True	Flag data based on expected RMS of post-fit spectra
<b>pofre_thresh</b>	double	1.3333	Threshold for expected RMS of post-fit spectra flag
<b>flag_prfr</b>	Bool	True	Flag data based on RMS of pre-fit spectra
<b>prfr_thresh</b>	double	4.5	Threshold for RMS of pre-fit flag
<b>flag_pofr</b>	Bool	True	Flag data based on RMS of post-fit spectra
<b>pofr_thresh</b>	double	4.0	Threshold for RMS of post-fit spectra flag
<b>flag_prfrm</b>	Bool	True	Flag data based on running mean of pre-fit spectra
<b>prfrm_thresh</b>	double	5.5	Threshold for running mean of pre-fit spectra flag
<b>prfrm_nmean</b>	int	5	Number of channel for running mean of pre-fit spectra flag
<b>flag_pofrm</b>	Bool	True	Flag data based on running mean of post-fit spectra
<b>pofrm_thresh</b>	double	5.0	Threshold for running mean of post-fit spectra flag
<b>pofrm_nmean</b>	Int	5	Number of channel for running mean of post-fit spectra flag
<b>plotflag</b>	Bool	True	Create plots for flagging
<b>infiles</b>	StringArray	None	List of input files to be calibrated
<b>spw</b>	String	None	Set of data selection by spectral windows

<b>pol</b>	String	None	Set of data selection by polarizations
<b>dryrun</b>	Bool	False	Run the task (False) or display task (True)
<b>acceptresults</b>	Bool	True	Add the results into the pipeline context

## 8.4 *hsd\_calsky*

hsd\_calsky generates calibration tables that contain a collection of spectra of the OFF position. For science SBs, the OFF position can be distinguished in Pipeline with an intent of "OFF\_SOURCE". For the case of the AmpCal SB, the edges of the raster rows are regarded as OFF-source position. See Fig. 2 for an illustration of this.

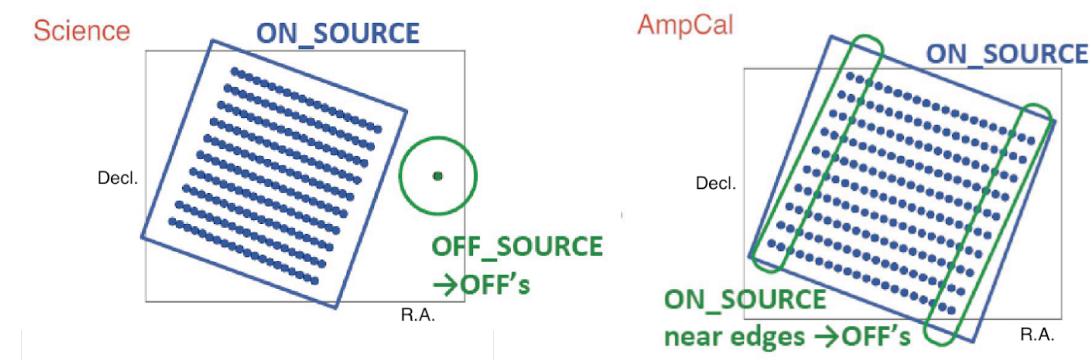


Fig. 2 Schematic view of Telescope Pointing. The Pipeline can handle two types of OFF-source positions. If calmode='ps' (for science SB), the Pipeline uses the spectra on the defined OFF-source positions. If calmode='otf' (for amplitude calibrator SB), the Pipeline generate the reference spectra using the scans at the edge of the map.

### Task Description

Generate a caltable for sky calibration that stores reference spectra, which is to be subtracted from on-source spectra to filter out other signals than the science target, such as sky background and signal from the receiver.

#### ---- pipeline parameters that can be set in any pipelinemode

**calmode** -- Calibration mode. Available options are 'auto' (default), 'ps', 'otf', and 'otfraster'. When 'auto' is set, the task will use the calibration mode defined by hsd\_inspectdata. 'ps' mode is a simple position switching mode using explicit reference scans. When calmode='otf' or 'otfraster', the pipeline will generate reference data from scans at the edge of the map. These modes are intended for OTF observations. The 'otf' is defined for generic scanning pattern such as Lissajous, while the 'otfraster' is specifically used for raster scan.

default: 'auto'

options: 'auto', 'ps', 'otf', 'otfraster'

**fraction** -- Subparameter for calmode, when 'otf' or 'otfraster' is set in calmode. Specify how much percentage of the map from one edge should be used as OFF-source scans.

default: '10%'

options: String style like '20%', or float value less than 1.0. For 'otfraster' mode, you can also

specify 'auto'.

**noff** -- Subparameter for calmode, when 'otfraster' is set in calmode. Specify the number of OFF scans near the edge, instead of using 'fraction'. If noff is set, the value in noff is used rather than values of 'fraction'.

default: -1 (use setting by 'fraction')

options: any positive integer value

**width** -- Subparameter for calmode, when 'otf' is set in calmode. Specify pixel width with respect to a median spatial separation between neighboring two data in time. Default will be fine in most cases.

default: 0.5

options: any float value

**elongated** -- Subparameter for calmode, when 'otf' is set in calmode. Set True only if observed area is elongated in one direction.

default: False

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.

default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**infiles** -- List of data files. The data file names must be those of Scantables (i.e. files with \*.asap suffix) that are registered to context via the hsd\_mstoscantable task.

default: [] (=all data in context)

**field** -- The list of field name(s). Defaults to all fields in context.

default: " (=all fields)

**spw** -- The list of spectral window(s) to be processed. Defaults to all science spectral windows.

default: " (=all spectral windows)

example: '3,4' (generate caltable for spw 3 and 4)

['0','2'] (spw 0 for first data, 2 for second)

**scan** -- The list of scan number(s) to be processed. Defaults to all scan numbers.

default: " (=all scans)

example: '22,23' (use scan 22 and 23 for calibration)

['22','24'] (scan 22 for first data, 24 for second)

**pol** -- The list of polarizations to be processed. Defaults to all polarizations.

default: " (=all polarizations)

example: '0' (generate caltable for pol 0)

['0~1','0'] (pol 0 and 1 for first data, only 0 for second)

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Examples

1. Generate caltables for all data managed by context.

```
hsd_calsky(pipelinemode='automatic')
```

### Parameter List

**Table 49: hsd\_calsky default settings**

Parameter	Type	Default	Description
<b>calmode</b>	String	'auto'	Calibration mode
<b>fraction</b>	String, or Double	'10%' (=0.1)	Subparameter for calmode='otf' or 'otfraster'. fraction of the OFF data to mark.
<b>noff</b>	Int	-1	Subparameter for calmode = 'otfraster'. number of the OFF data to mark.
<b>width</b>	Double	0.5	Subparameter for calmode = 'otf'. width of the pixel for edge detection.
<b>elongated</b>	Bool	False	Subparameter for calmode = 'otf'. Whether observed area is elongated in one direction or not
<b>edge</b>	IntArray	None	Number of edge channels to be excluded from statistic calculation to flag the data
<b>infiles</b>	StringArray	None	List of input files to be calibrated
<b>field</b>	String	None	Field to be calibrated
<b>spw</b>	String	None	Set of data selection by spectral windows
<b>scan</b>	String	None	Set of data selection by scan numbers
<b>pol</b>	String	None	Set of data selection by polarizations
<b>dryrun</b>	Bool	False	Run the task (False) or display task (True)
<b>acceptresults</b>	Bool	True	Add the results into the pipeline context

## ***8.5 hsd\_exportdata***

*hsd\_exportdata* is to export the image cube defined in the Pipeline context to the data products directory, converting and/or packing if necessary. The current version exports the following products for the Science SB:

- an XML file containing the Pipeline processing request
- a tar file per ASDM/MS containing the final flag-version files
- a text file per ASDM/MS containing a list of CASA commands to apply calibration tables
- a FITS image for each selected Science target source image
- a tar file per MS containing caltables for the MS
- a tar file containing the Weblog,
- a text file containing the final list of CASA commands
- a python script that can be used to calibrate the data
- a text file containing the Jansky-to-Kelvin factors

### **Task Description**

Export the data defined in the pipeline context to the data products directory, converting and or packing it as necessary.

#### **---- pipeline parameters that can be set in any pipelinemode**

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

#### **---- pipeline-defined parameters which can be set only in 'interactive mode'**

**pprfile** -- Name of the pipeline processing request to be exported. Defaults to a file matching the template 'PPR\_\*.xml'.  
default: [] (none)  
example: pprfile=['PPR\_GRB021004.xml']

**targetimages** -- List of science target images to be exported. Defaults to all science target images recorded in the pipeline context.  
default: [] (none)  
example: targetimages=['r\_aqr.CM02.spw5.line0.XXYY.sd.im',  
'r\_aqr.CM02.spw5.XXYY.sd.cont.im']

**products\_dir** -- Name of the data products subdirectory.  
default: "  
example: products\_dir='..../products'

#### **--- pipeline task execution modes**

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute

(True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

## Examples

1. Export the pipeline results for a single sessions to the data products directory

```
!mkdir ..../products  
hsd_exportdata (products_dir='..../products')
```

## Parameter List

**Table 50: hsd\_exportdata default settings**

Parameter	Type	Default	Description
<b>pprfile</b>	String	None	The pipeline processing request (PPR) file to be exported
<b>targetimages</b>	StringArray	None	List of target CASA images to be exported
<b>products_dir</b>	String	None	The data products directory
<b>dryrun</b>	Bool	False	Run the task (False) or display task command (True)
<b>acceptresults</b>	Bool	True	Add the results into the pipeline context

## 8.6 *hsd\_flagdata*

This task applies online flags, a flagging templates (manual flagging provided by user), flags of shadowed data, flag of edge channels, identification and flagging of unwanted intents such as POINTING, FOCUS, SIDEBAND, and ATMOSPHERE. Note that Cycle 1 and Cycle 2 Data with the DelayCal (delay calibrator) intent is also flagged at this step, since having delay cal in Total Power observations is just to invoke pointing and focus observations, and not to be used in data reduction.

### Task Description

Do basic flagging of a list of measurement sets

---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

**autocorr** -- Flag autocorrelation data.  
default: False

**shadow** -- Flag shadowed antennas.  
default: True

**scan** -- Flag a list of specified scans.  
default: True

**scannumber** -- A string containing a comma delimited list of scans to be flagged.  
example: '3,5,6'  
default: "

**intents** -- A string containing a comma delimited list of intents against which the scans to be flagged are matched.  
example: "\*BANDPASS"  
default: 'POINTING,FOCUS,ATMOSPHERE,SIDEband'

**edgespw** -- Flag the edge channels of spectral windows.  
default: True

**fracspw** -- Fraction of edge channels to be flagged on the TDM-mode spectral window (for BaseLine correlator).  
default: 0.0625

**fracspwfps** -- Fraction of edge channels to be flagged on the TDM-mode spectral window (for ACA correlator).  
default: 0.48387

**online** -- Apply the online flags.  
default: True

**fileonline** -- Name of a file containing the online flags. The online flags are computed by the h\_init or hif\_importdata data tasks. If the online flags files are undefined, a name of the form '*msname\_flagonline.txt*' is assumed.  
default: "

**template** -- Apply flagging templates  
default: True

**filetemplate** -- The name of an text file that contains the flagging template for RFI, birdies, telluric lines, etc. If the flag template files are undefined, a name of the form

'msname\_flagtemplate.txt' is assumed.

default: "

**hm\_tbuff** -- The heuristic for computing the default time-interval-padding parameter. The options are 'halfint' and 'manual'. In 'halfint' mode tbuff is set to half the maximum of the median integration time of the science- and calibrator-target observations.

default: 'halfint'

**tbuff** -- The time in seconds used to pad flagging command time intervals if hm\_tbuff='manual'.  
default: 0.0

**qa0** – QA0 flags

default: True

### **--- pipeline-defined parameters which can be set only in 'interactive mode'**

**vis** -- The list of input measurement sets. Defaults to the list of measurement defined in the pipeline context.

default: "

example: vis=['X227.ms', 'X124.ms']

**flagbackup** -- Back up any pre-existing flags.

default: False

### **--- pipeline task execution modes**

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

### **Output:**

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### **Example**

1. Do basic flagging on a measurement set

hsd\_flagdata(pipelinemode='automatic')

2. Do basic flagging on a measurement set, flagging additional scans selected by number as well.

hsd\_flagdata(scan=True, scannumber='13,18')

### **Parameter List**

**Table 51: hsd\_flagdata default settings**

Parameter	Type	Default	Description
<b>autocorr</b>	Bool	False	Flag autocorrelation data
<b>shadow</b>	Bool	True	Flag shadowed antennas
<b>scan</b>	Bool	True	Flag specified scans
<b>scanumber</b>	String	None	List of scan numbers to be flagged
<b>intents</b>	String	'POINTING,FOCUS,ATMOSPHERE,SIDEband,CHANNEL,ECK'	List of intents of scans to be flagged
<b>edgespw</b>	Bool	True	Flag edge channels
<b>fracspw</b>	String	'1.875GHz'	Fraction of baseline correlator edge channels to be flagged
<b>fracspwfps</b>	Double	0.048387	Fraction of ACA correlator edge channels to be flagged
<b>online</b>	Bool	True	Apply the online flags
<b>fileonline</b>	String	None	File of online flags to be applied
<b>template</b>	Bool	True	Apply a flagging template
<b>filemplate</b>	String	None	File that contains the flagging template
<b>hm_tbuff</b>	String	'halfint'	The time buffer computation heuristic
<b>qa0</b>	Bool	True	QA0 flags
<b>vis</b>	StringArray	None	List of input MSes to flag
<b>flagbackup</b>	Bool	False	Backup preexistings flags before applying new ones
<b>dryrun</b>	Bool	False	Run the task (False) or display the command (True)
<b>acceptresults</b>	Bool	True	Add the results into the pipeline context

## 8.7 *hsd\_imaging*

*hsd\_imaging* applies flagging tables created by *hsd\_bflag*, converts from Scantables to MS format, simply applies pre-computed Kelvin-to-Jy conversion factor to the Science data, and then does imaging with the native spectral resolution. Single Dish images per antenna are also created as well as combined images including all antennas.

Note that images generated here are always in LSRK frame. All imaging parameters are by default set to cell size of 1/9 of theoretical beam size, and grid function of Spheroidal Function (SF). The user cannot be re-set in pipeline.

### Note on Kelvin-to-Jy conversion factor

The Pipeline is NOT designed to derive the Kelvin-to-Jy conversion factor (K-to-Jy factor), but just to apply it. The factor needs to be derived manually, and the information (K-to-Jy factor for each science EB, spectral window, polarization, and antenna) should be put in a file, which will serve as input to this task. We recommend to use the file name, "jyperk.csv", since the *hsd\_imaging* task find a text file with that name by default. If the K-to-Jy factors are successfully applied, the Weblog gives a list of those factors in the summary page. Note that in case there is no "jyperk.csv" or no other files have been specified as input, the Weblog will give a warning, but the Pipeline will not abort due to this.

**mode** -- Imaging mode controlling imaging parameters in the task. Accepts either 'line' (spectral line imaging) or 'ampcal' (image settings for amplitude calibrator)  
default: 'line'  
options: 'line', 'ampcal'

**refile** -- Path to a file containing K-to-Jy factors for science data, which must be manually provided. K-to-Jy factor must take into account all efficiencies, i.e., it must be a direct conversion factor from  $T_a(K)$  to Jy. The file must be in either MS-based or session-based format. The MS-based format must be in an CSV format with five fields: MS name, antenna name, spectral window id, polarization string, and Jy/K conversion factor.

Example for the file is as follows:

```
MS,Antenna,Spwid,Polarization,Factor  
uid____A002_X316307_X6f.ms,CM03,5,XX,10.0  
uid____A002_X316307_X6f.ms,CM03,5,YY,12.0  
uid____A002_X316307_X6f.ms,PM04,5,XX,2.0  
uid____A002_X316307_X6f.ms,PM04,5,YY,5.0
```

The first line in the above example is a header which may or may not exist.

An example for the session-based format is as follows:

```
#OUSID=XXXXXX
```

```

#OBJECT=Uranus

#FLUXJY=yy,zz,aa

#FLUXFREQ=YY,ZZ,AA

#sessionID,ObservationStartDate(UTC),ObservationEndDate(UTC),Antenna,BandCenter(MHz),
BandWidth(MHz),POL,Factor

1,2011-11-11 01:00:00,2011-11-11 01:30:00,CM02,86243.0,500.0,I,10.0

1,2011-11-11 01:00:00,2011-11-11 01:30:00,CM02,86243.0,1000.0,I,30.0

1,2011-11-11 01:00:00,2011-11-11 01:30:00,CM03,86243.0,500.0,I,50.0

1,2011-11-11 01:00:00,2011-11-11 01:30:00,CM03,86243.0,1000.0,I,70.0

1,2011-11-11 01:00:00,2011-11-11 01:30:00,ANONYMOUS,86243.0,500.0,I,30.0

1,2011-11-11 01:00:00,2011-11-11 01:30:00,ANONYMOUS,86243.0,1000.0,I,50.0

2,2011-11-13 01:45:00,2011-11-13 02:15:00,PM04,86243.0,500.0,I,90.0

2,2011-11-13 01:45:00,2011-11-13 02:15:00,PM04,86243.0,1000.0,I,110.0

2,2011-11-13 01:45:00,2011-11-13 02:15:00,ANONYMOUS,86243.0,500.0,I,90.0

2,2011-11-13 01:45:00,2011-11-13 02:15:00,ANONYMOUS,86243.0,1000.0,I,110.0

```

The line starting with '#' indicates a meta data section and header. The header must exist. The lines with the Antenna name of 'ANONYMOUS' are used when there is no measurement for specific antenna in the session. In the above example, if science observation of session 1 contains the antenna PM04, K-to-Jy factor for ANONYMOUS antenna will be applied since there is no measurement for PM04 in session 1. If no file name is given or the specified file here doesn't exist, all K-to-Jy factors are set to 1.0.

default: 'jyperk.csv'

example: ", 'working/jyperk.csv'

## Task Description

Apply flagging and K-to-Jy factors on science data

Generate Single Dish Images for individual antenna as well as combined image

### ---- pipeline parameters that can be set in any pipelinemode

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**infiles** -- List of data files. The data file names must be those of Scantables (i.e. files with \*.asap suffix) that are registered to context via hsd\_mstoscantable task.

default: [] (all data in context)

**field** – The list of field name(s) to be processed.  
default: " (=all fields)

**spw** -- The list of spectral window(s) to be processed. Defaults to all science spectral windows.  
default: " (=all spectral windows)

**scan** – The list of scan number(s) to be processed. Defaults to all scan numbers.  
default: " (=all scans)

**pol** – The list of polarization(s) to be processed. Defaults to all polarizations.  
default: " (=all polarizations)

### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Example

1. Do basic imaging for a measurement set  
`hsd_imaging(pipelinemode='automatic')`

### Parameter List

**Table 52: hsd\_imaging default settings**

Parameter	Type	Default	Description
<b>mode</b>	String	'line'	Imaging mode ['line' 'ampcal']
<b>reffeile</b>	String	'jyperk.csv'	File of Jy/K conversion factor
<b>infiles</b>	StringArray	None	List of input files to be imaged
<b>field</b>	String	None	Field to be imaged
<b>spw</b>	String	None	Set of data selection by spectral windows
<b>scan</b>	String	None	Set of data selection by scan numbers
<b>pol</b>	String	None	Set of data selection by polarizations
<b>dryrun</b>	Bool	False	Run the task (False) or display task (True)

<b>acceptresults</b>	Bool	True	Add the results into the pipeline context
----------------------	------	------	---

## 8.8 *hsd\_importdata*

This task loads the ASDM into the pipeline context, unpacking and converting it to Measurement Set (MS) format, similarly to the interferometry task hifa\_importdata (hsd\_importdata actually shares the code with hifa\_importdata task). In addition, it generates two flagging files, one for online flagging commands (\*flagonline.txt) and the other for manual flagging (\*flagtemplate.txt).

### Task Description

Import data into pipeline.

#### ---- pipeline parameters that can be set in any pipelinemode

**vis** -- List of visibility data files. These may be ASDMs, tar files of ASDMs, MSs, or tar files of MSs, If ASDM files are specified, they will be converted to MS format.

default: [] (none)

example: vis=['X227.ms', 'asdms.tar.gz']

**session** -- List of sessions to which the visibility files belong. Defaults to a single session containing all the visibility files, otherwise a session must be assigned to each vis file.

default: [] (none)

example: session=['Session\_1', 'Sessions\_2']

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.

default: 'automatic'.

**lazy** -- Use the lazy filter import. If lazy=True, the imported MS does not have DATA column, but does a reference to the ASDM. It saves the disk space and processing time.

default: False

**dbservice** -- Use online flux catalog on import

default: True

**with\_pointing\_correction** -- add (ASDM::Pointing::encoder -

ASDM::Pointing::pointingDirection) to the value to be written in MS::Pointing::direction

default: True

**createmms** -- Create a multi-MS partitioned according to the given separation axis. For more detailed documentation on partition, Multi-MS and the MPI use in CASA, please see the help

partition and help mstransform.

default: 'automatic'

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**asis** – Creates verbatim copies of the ASDM tables in the output measurement set. Value must be given as a string of table names separated by spaces. A wildcard (\*) is allowed to use.

default: 'Antenna Station Receiver CalAtmosphere'

example: 'Receiver', "

**process\_caldevice** -- Ingest the ASDM caldevice table

default: False

**overwrite** -- Overwrite existing measurement sets on output.

default: False

**bdfflags** – Apply the ASDM binary data format flags to output measurement set.

default: True

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).

default: True

#### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

#### Examples

1. Load an ASDM list in the .../rawdata subdirectory into the context.

```
hsd_importdata (vis=['..../rawdata/uid____A002_X30a93d_X43e',
'..../rawdata/uid_A002_x30a93d_X44e'])
```

2. Load an MS in the current directory into the context.

```
hsd_importdata (vis=[uid____A002_X30a93d_X43e.ms])
```

3. Load a tarred ASDM in .../rawdata into the context.

```
hsd_importdata (vis=['..../rawdata/uid____A002_X30a93d_X43e.tar.gz'])
```

4. Check the hsd\_importdata inputs, then import the data

```
myvislist = ['uid__A002_X30a93d_X43e.ms', 'uid_A002_x30a93d_X44e.ms']
hsd_importdata(vis=myvislist, pipelinemode='getinputs')
hsd_importdata(vis=myvislist)
```

5. Load an ASDM but check the results before accepting them into the context.

```
results = hsd_importdata (vis=['uid__A002_X30a93d_X43e.ms'], acceptresults=False)
results.accept()
```

6. Run in dryrun mode before running for real

```
results = hsd_importdata (vis=['uid__A002_X30a93d_X43e.ms'], dryrun=True)
results = hsd_importdata (vis=['uid__A002_X30a93d_X43e.ms'])
```

## Parameter List

Table 53: hsd\_importdata default settings

Parameter	Type	Default	Description
<b>vis</b>	StringArray	None	List of input visibility file
<b>session</b>	StringArray	None	List of visibility data sessions
<b>asis</b>	String	'Antenna Station Receiver CalAtmosphere CalWVR'	Create verbatim copies of the ASDM tables in the output MS.
<b>process_caledevice</b>	Bool	False	Import the caldevice table from the ASDM
<b>overwrite</b>	Bool	False	Overwrite existing files on import
<b>bdfflags</b>	Bool	True	Apply BDF flags on import
<b>dryrun</b>	Bool	False	Run the task (False) or display task (True)
<b>acceptresults</b>	Bool	True	Add the results into the pipeline context
<b>lazy</b>	Bool	False	Use the lazy import option
<b>dbservice</b>	Bool	True	Use the online flux catalog
<b>with_pointing_correction</b>	Bool	True	Apply pointing correction to DIRECTION
<b>createmms</b>	String	'automatic'	Create a Multi-MS

## **8.9 *hsd\_inspectdata***

The Pipeline generates plots of Telescope Pointing, Azimuth and Elevation, Weather, and WVR. This task also groups spectral window IDs for different EBs, and also does matching between Tsys and science spectral windows (Tsys mapping).

### **Note on Grouping of spectral windows in different EBs**

When several EBs are processed at once, the Pipeline needs to group their respective science spectral windows, so that the Pipeline processes the grouped spectral windows together for the different EBs. The grouping is done based on spectral window NAME.

### **Note on Tsys mapping**

Even though the Tsys mapping is done in hifa\_tsystcal, hsd\_inspectdata task also does the matching between Tsys spectral windows and Science spectral windows. This is because this task was inherited from the old procedure that did not have hifa\_tsystcal. Specially in case of processing Cycle1 and Cycle2 data in pipeline, users are recommended to confirm that the Tsys mapping are identical between the one in hsd\_inpspectdata and the one in hifa\_tsystcal.

### **Task Description**

Inspect whole data registered to the pipeline context.

#### **---- pipeline parameters that can be set in any pipelinemode**

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'

#### **---- pipeline-defined parameters which can be set only in 'interactive mode'**

**infiles** -- List of data files. The data file names must be those of Scantables (i.e. files with \*.asap suffix) that are registered to context via hsd\_mstoscantable task.  
default: [ ] (none)

#### **--- pipeline task execution modes**

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

#### **Output:**

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

## Examples

1. Import data via hsd\_importdata task and inspect them.

```
hsd_importdata(vis=['uid__A002_X30a93d_X43e','uid_A002_x30a93d_X44e'])  
hsd_inspectdata()
```

2. Import data via hsd\_importdata task and inspect some of them

```
hsd_importdata(vis=['uid__A002_X30a93d_X43e', 'uid_A002_x30a93d_X44e'])  
hsd_mstoscantable(infiles=['uid__A002_X30a93d_X43e.ms',  
'uid__A002_X30a93d_X44e.ms'])  
hsd_inspectdata(infiles=['uid__A002_X30a93d_X43e.PM01.asap',  
'uid__A002_X30a93d_X44e.PM01.asap'])
```

## Parameter List

**Table 54: hsd\_inspectdata default settings**

Parameter	Type	Default	Description
<b>infiles</b>	StringArray	None	List of input files to be inspected
<b>dryrun</b>	Bool	False	Run the task (False) or display task (True)
<b>acceptresults</b>	Bool	True	Add the results into the pipeline context

## 8.10 hsd\_mstoscantable

hsd\_mstoscantable converts the measurement set (MS) format files into Scantable format for each antenna. Also, it calculates averaged Tsys over the science frequency ranges using the Tsys table generated in hifa\_tsyscal, and generates new Tsys tables so that it is able to be applied to the Scantable formatted file.

### Task Description

Convert MS- into Scantable- format files

#### ---- pipeline parameters that can be set in any pipelinemode

**infiles** -- List of data files. These data files may be ASDMs, tar files of ASDMs, MSs, tar files of MSs, Scantables, or other types of single dish data that can be imported by CASA (ATNF SDFITS, RPFITS, NRO raw data). Whole data will be converted to Scantable format. If ASDM files are specified, they will be converted to MS format first, then converted to Scantable format. default: [] (all data in context)

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

---- **pipeline-defined parameters which can be set only in 'interactive mode'**

**overwrite** -- Overwrite existing MSs on output.  
default: True

--- **pipeline task execution modes**

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

**Output:**

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

**Parameter List**

**Table 55: hsd\_mstoscantable default settings**

Parameter	Type	Default	Description
<b>infiles</b>	StringArray	None	List of input files
<b>overwrite</b>	Bool	True	Overwrite existing files on import
<b>dryrun</b>	Bool	False	Run the task (False) or display task (True)
<b>acceptresults</b>	Bool	True	Add the results into the pipeline context

## **8.11 hsd\_plotflagbaseline**

hsd\_plotflagbaseline is to generate plots before and after baseline subtraction for every single spectrum, and to summarize latest baseline subtraction and flagging.

**Task Description**

Generate plots to summarize latest baseline subtraction and flagging

---- **pipeline parameters that can be set in any pipelinemode**

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**infiles** -- List of data files. The data file names must be those of Scantables (i.e. files with \*.asap suffix) that are registered to context via hsd\_mstoscantable task.  
default: [] (=all data in context)

**spw** – The list of spectral windows to be processed. Defaults to all science spectral windows.  
default: " " (=all spectral windows)

**pol** – The list of polarization to be processed. Defaults to all polarizations.  
default: " " (=all polarizations)

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).  
default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

#### Output:

results -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

#### Parameter List

**Table 56: hsd\_plotflagbaseline default settings**

Parameter	Type	Default	Description
<b>infiles</b>	StringArray	None	List of input files to be calibrated
<b>spw</b>	String	None	Set of data selection by spectral windows
<b>pol</b>	String	None	Set of data selection by polarizations
<b>dryrun</b>	Bool	False	Run the task (False) or display task (True)

<b>acceptresults</b>	Bool	True	Add the results into the pipeline context
----------------------	------	------	---

## 8.12 *hsd\_simplescale*

*hsd\_simplescale* is used to multiply the spectra (in scantable) by a scaling factor (1.25) to Cycle 1 and 2 data only to fix the ACA correlator non-linearity issue. This task was dropped from the Cycle 3 Single Dish Pipeline standard procedure after upgrading the ACA correlator.

### Task Description

Multiply spectra in scantables by a scaling factor.

#### ---- pipeline parameters that can be set in any pipelinemode

**factor** -- A scaling factor to apply.

default: 1.25

**pipelinemode** -- The pipeline operating mode. In 'automatic' mode the pipeline automatically determines the values of all pipeline context to pipeline-defined parameters. In 'interactive' mode the user can manually set the values to user-defined parameters. In 'getinputs' mode the user can check the settings of all parameters in the task without running the task.  
default: 'automatic'.

#### ---- pipeline-defined parameters which can be set only in 'interactive mode'

**infiles** -- List of data files. The data file names must be those of Scantables (i.e. files with \*.asap suffix) that are registered to context via *hsd\_mstoscantable* task.

default: [] (all data in context)

#### --- pipeline task execution modes

**dryrun** -- Run the command (False) or generate the commands to be run but do not execute (True).

default: False

**acceptresults** -- Add the results of the task to the pipeline context (True) or reject them (False).  
default: True

### Output:

**results** -- If pipelinemode is 'getinputs' then None is returned. Otherwise the results object for the pipeline task is returned.

### Parameter List

**Table 57: *hsd\_simplescale* default settings**

Parameter	Type	Default	Description
<b>factor</b>	Double	1.25	A scaling factor to apply

<b>infiles</b>	StringArray	None	List of input scantables
<b>dryrun</b>	Bool	False	Run the task (False) or display task (True)
<b>acceptresults</b>	Bool	True	Add the results into the pipeline context



The Atacama Large Millimeter/submillimeter Array (ALMA), an international astronomy facility, is a partnership of Europe, North America and East Asia in cooperation with the Republic of Chile. ALMA is funded in Europe by the European Organization for Astronomical Research in the Southern Hemisphere (ESO), in North America by the U.S. National Science Foundation (NSF) in cooperation with the National Research Council of Canada (NRC) and the National Science Council of Taiwan (NSC) and in East Asia by the National Institutes of Natural Sciences (NINS) of Japan in cooperation with the Academia Sinica (AS) in Taiwan. ALMA construction and operations are led on behalf of Europe by ESO, on behalf of North America by the National Radio Astronomy Observatory (NRAO), which is managed by Associated Universities, Inc. (AUI) and on behalf of East Asia by the National Astronomical Observatory of Japan (NAOJ). The Joint ALMA Observatory (JAO) provides the unified leadership and management of the construction, commissioning and operation of ALMA.



