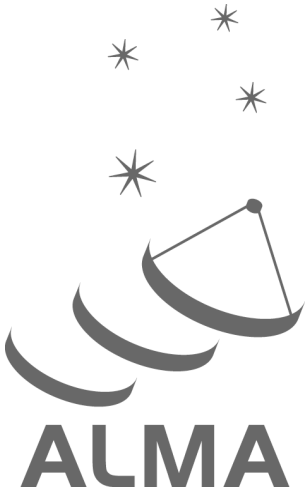
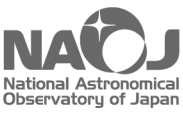
Doc 4.13, ver. 2 | July, 2017

ALMA Science Pipeline User's Guide for CASA 4.7.2

Interferometric and Single-Dish Processing





www.almascience.org

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**. Helpdesk tickets will be directed to the appropriate ALMA Regional Center at ESO, NAOJ or NRAO.

Revision History:

Version	Date	Editors
3.13v1.0 CASA 4.5.1	January 2016	Pipeline Team
4.13v1.0 CASA 4.7.0	October 2016	Pipeline Team
4.13v2.0 CASA 4.7.2	July 2017	Pipeline Team

In publications, please refer to this document as:

ALMA Pipeline Team, 2017, ALMA Science Pipeline User's Guide, ALMA Doc 4.13v2.0

Table of contents

1 The ALMA Science Pipeline

1.1 Purpose of this document

This document describes how to obtain the ALMA Pipeline, how to use it to calibrate and image ALMA interferometric (IF) and single-dish (SD) data, and a description of the Pipeline WebLog (collection of web pages with diagnostic information describing the pipeline run). Since interferometric and single-dish data are calibrated and imaged using different procedures and diagnostics, their recalibration procedures and WebLogs are described separately.

This document is applicable for the Cycle 4 “patch” version of the ALMA Pipeline that is included in CASA 4.7.2, deployed for use in ALMA Operations in April 2017. This version is labeled as Pipeline Version r39732 (Pipeline-Cycle4-R2-B) CASA Version 4.7.2 r39762 in the WebLog.

1.2 Pipeline Overview and nomenclature

The ALMA Science Pipeline is used for the automated calibration and imaging of ALMA interferometric and single-dish data. ALMA Interferometric data refers to observations obtained with either the ALMA 12-m Array or 7-m Array, while single-dish data refers to observations obtained with the 12-m dishes of the ALMA Total Power Array.

The Pipeline consists of modular calibration and imaging tasks within the Common Astronomy Software Applications (CASA) data reduction package that are selected and put together in a specific order based on standard prescriptions or **recipes**. The ALMA pipeline recipes cover the processing requirements of ALMA “standard” interferometric and single-dish observing modes. Datasets resulting from ALMA “non-standard” observing modes are, as a rule, processed outside the pipeline, using manually modified CASA scripts. The standard and non-standard observing modes are defined in the Proposer’s Guide for each ALMA proposal cycle. The science pipeline is not yet commissioned for the combination of datasets obtained from different array components (separate IF array observations, or IF plus SD combinations).

The pipeline operates on a completed dataset that is comprised of all of the quality assured individual executions that result from completing a Scheduling Block (SB). An individual SB execution results in a dataset referred to as an **ASDM** (for ALMA Science Data Model), and the collection of ASDMs from a single SB are collected into a data structure called a Member Observing Unit Sets (**MOUS**), which is the data unit that the pipeline operates on. The pipeline produces the following: calibration products for each ASDM (including calibration and flagging files and tables); imaging products (FITS images) made from all ASDMs (although not necessarily for all science targets); informative logs and scripts; and a **WebLog** consisting of a collection of webpages with diagnostic messages, tables, figures, and “Quality Assurance" (**QA**) scores. These products are reviewed as part of the ALMA Quality Assurance process, and, if satisfactory, are stored into the ALMA Science Archive. See the ALMA Technical Handbook for details on the ALMA data structures, quality assurance criteria, and archiving system.

The Pipeline is data-driven: i.e. the characteristics of each dataset drive the calibration and imaging strategy (the **Pipeline Heuristics**). During the Pipeline run, critical information (for example, which calibration tables are used) are stored in the pipeline **Context**. Both the **Heuristics** and the **Context** are implemented as python classes.

In order to determine if the Pipeline was used in the processing of an ALMA dataset, please look at the WebLog or consult the README file in the data delivery package. Some projects may contain a mix of both manually and Pipeline-calibrated data.

2 What's New in Cycle 4

2.1 What's New for the start of Cycle 4 (October 2016)

New features of the original Cycle 4 pipeline (released with CASA 4.7.0) include:

- The interferometric calibration pipeline has a low signal-to-noise heuristic task (`hifa_spwphaseup`) that calculates the temporal phase variations by combining spectral windows (spw).
- The Interferometry Pipeline includes science target imaging, imaging of "check sources" in the calibrator imaging stage, and improved calibrator quality assurance scores.
- The Single-dish Pipeline was refactored to use Measurement Set (MS) rather than scantable format.
- There are improved defaults for the `hif_gainflag` task.
- Files previously exported as .tar.gz are now exported as .tgz.

2.2 What's New for the Cycle 4 "patch" (April 2017)

The primary new functionality of the Cycle 4 pipeline patch (released with CASA 4.7.2) includes:

- Exporting of the imaging products for "check source" calibrators.
- A new Image size mitigation task `hif_checkproductsize` that reduce the image product size if it is predicted to be larger than specific limits, implemented with the aim of keeping the processing time ≤ 3 days when run on a node with 128GB RAM. The mitigations are to limit channel binning, imaged field of view, number of image pixels per synthesized beam, and/or number of sources imaged. The default limits for triggering the mitigation are: cubes estimated to exceed 30 GB; total imaging products estimated to exceed 400 GB. These limits are a coarse surrogate for processing time.
- Improvements in IF WebLog pages:
  - New per-antenna plots in the applycal "details" page, to help identify the specific antennas responsible for outlier points.
  - A "next" functionality for reviewing the `hif_findcont` plots.
  - New table format in the `hifa_gfluxcale` WebLog page for calibrators that shows the cataloged values of flux as well as the derived flux values.
- Improvements in SD WebLog pages:
  - New plots for atmospheric transmission added to `hsd_baseline, hsd_imaging.`
  - Anchors added on the top page of each task.
  - New summary table in `hsd_blflag`.
  - New source and MS filters in the detail pages of `hsd_baseline` and `hsd_skycal`.
  - Online-flagged data points are now shown in gray in TP pointing plots.
  - Image captions added in the navigation plots in `hsd_imaging`.
  - New per-spw plots in hsd_skycal
- Fixed various bugs that led to errors in `hif_findcont`, WebLog plots, or because of odd source names.

2.3 Current Known Limitations of the Cycle 4 Pipeline

The current (April 2017) **Known limitations** of the Cycle 4 pipeline include:

- The pipeline is commissioned only for ALMA "standard mode" observations, as defined in the Proposers Guide for the latest cycle, subject to the additional restrictions listed below.
- All raw data (ASDMs) run through the pipeline must have complete and properly formatted binary and metadata. This is not always the case for ASDMs from earlier ALMA cycles. In particular:
  - The SD pipeline can only be run on data from Cycle 3 or later.
  - The IF pipeline will not work with ALMA Cycle 0 data, or with some Cycle 1 – 2 data.
- Manually calibrated data from Cycles 1 – 3 are likely to have problems if run through the pipeline.
- The raw data (ASDMs) run through the pipeline should have a "quality assurance level 0" (QA0) assessment of "QA0 Pass". Running the pipeline on non-quality assured data ("QA0 SemiPass" or "QA0 Fail") is not expected to give sound results and may fail.
- The pipeline assumes that it has access to all of the available RAM on the node where it is run. If other processes use signifcant amounts of this RAM, the pipeline may fail.
- If the number of files opened by the pipline (which increases with the number of ASDMs being processed) exceeds the server specification, the pipeline will fail. This limit can generally be increased using a command like "`ulimit –Sn 4096`"

Additional limitations of the Interferometric Pipeline are:

- The IF pipeline does not perform flux equalization between the different executions of multi-epoch observations.
- The IF pipeline can calibrate ephemeris target data, but will not properly image them.
- While the IF pipeline calibration and flagging tasks include low signal-to-noise heuristics, they will produce poor results if the calibrators are too weak.
- The IF pipeline does not perform automated science target flagging. Template flagging files (names like `uid*_flagtargetstemplate.txt`, one per ASDM) are provided for users to add their own flags; these will be applied during the hifa_flagtargets task if the pipeline imaging script is re-run.
- In order to increase delivery rates of data to PIs, the archived imaging products may be binned in frequency, limited in the imaged field of view, and/or restricted to a subset of sources. Users can make the missing products by making small modifications to the scripts that are archived with the data.
- The IF imaging pipeline will try to image all field pointings associated with a given source name. If these are too widely spaced the resulting images will be very large and sparse. Such poorly formatted "mosaics" should be imaged manually instead.
- The frequency ranges for interferometric continuum identification and subtraction are done in an automated manner that works well over a very broad range of observing modes and source properties. In some cases (e.g. hot core line emission, noisy broadband continuum), it is expected that better results can be obtained by more careful examination of individual sources and/or spectral windows. If the data are heavily binned in frequency before this task is run, the results may be compromised.

- The IF continuum identification task `hif_findcont` uses various heuristics that may depend on the amount of available RAM on the cluster node. Running the corresponding pipeline task (hif_findcont) using a machine with a different amount of RAM than was used to produce the delivered products may therefore lead to the selection of different continuum ranges. However, the original products will be reproduced exactly using the pipeline-provided "cont.dat" file, which also significantly reduces the processing time.
- Science target deconvolution ("cleaning") is done with a generic mask and shallow dynamic-range limited clean thresholds, meaning that images with moderate to strong emission will benefit from more carefully defined masks and deeper cleaning thresholds.
- The IF PL imaging steps use the "effective channel bandwidth" from the raw data file to calculate the theoretical image sensitivity and hence clean thresholds. This information is not correctly entered for ALMA data from Cycles 2 and earlier; as a result, the clean thresholds will be higher than intended when such data is run through the imaging pipeline.
- The pipeline does not include science target self-calibration. Therefore, the pipeline imaging products of bright sources will be dynamic range limited.
- The interferometric imaging pipeline commands should work with measurement sets calibrated outside the pipeline, but this has not been tested extensively and may have as-yet undetermined failure modes.

Additional limitations of the Single-dish Pipeline are:

- The number and total size of all ASDMs run through the SD pipeline cannot exceed limits set by the server specification (amounting to 31 GB of raw data for a node with 64 GB of RAM).
- The SD pipeline does not work properly with ephemeris targets or targets with a high proper motion (> 200 μas over an execution).
- The SD pipeline does not support single-polarization data.
- The frequency ranges for single dish line identification and spectral baseline subtraction are done in an automated manner that has been optimized to detect moderate channel width (wider than 100 channels) emission lines at the center of a spectral window. It is expected that better results can be obtained by more careful examination of individual sources and/or spectral windows. The following cases are most strongly affected:
  - Narrow emission lines (less than 100 channels wide), especially in TDM mode.
  - Emission at the edge of a spectral window.
  - Cubes with a "forest" of emission lines.
- The SD pipeline imaging results may be unusable if there is emission in the "off" position.
- The SD pipeline will not run correctly on multi-execution datasets in which some sources are missing from some of the executions.
- When running **scriptForPI.py** scripts with versions of v1.19 or earlier, you may get an error message of the form: "ERROR: uid___A002_XXXX_XXXX.PM*.ms was not created." This message can be ignored.

A list of pipeline "known issues" that arise after the publication date of this document is maintained on the ALMA Science Portal at http://almascience.org/processing/science-pipeline#KI. This list will be updated as issues are discovered during the cycle.

3 Pipeline Versions & Documentation

3.1 Obtaining the Pipeline

A link to the version of CASA 4.7.2 that includes the ALMA pipeline is available, along with installation instructions and supporting documentation, from the **Science Pipeline** section of the **ALMA Science Portal** at http://www.almascience.org (under the "Processing" tab, or directly at http://almascience.org/processing/science-pipeline). If any issues are encountered with CASA 4.7.2 installation, please contact the ALMA Helpdesk via the link on the ALMA Science Portal.

The pipeline tasks become availalble by starting up CASA using the command:

```
%casapy --pipeline
```

3.2 Pipeline-related Documentation

The User documentation currently relating to the Pipeline is also available from the from the Science Pipeline section of the Science Portal referenced above. This includes:

- **ALMA Science Pipeline User's Guide**: This document.
- **ALMA Science Pipeline Reference Manual**: Description of individual Pipeline tasks.

In addition, Chapter 13 of the **ALMA Technical Handbook** briefly describes ALMA pipeline processing, and the **ALMA QA2 Data Products** document for Cycle 4 provides a description of ALMA data deliveries, including pipeline products.

Finally, examples of common re-imaging modifications to the IF pipeline script are given at: https://casaguides.nrao.edu/index.php/ALMA_Imaging_Pipeline_Reprocessing.

3.3 Pipeline & CASA Versions

The pipeline heuristic tasks have a specific version number, and are bundled with a specific version of CASA. These versions are reported in the README file that is archived with the pipeline data products, and are also reported on the Home page of the WebLog for each pipeline-processed dataset (see Figure 9 for an example). For ALMA Cycle 4, the pipeline was originally released with CASA 4.7.0 in October 2016, and the current "Cycle 4 patch" version was released with CASA 4.7.2 in April 2017.

In general, the version of Pipeline+CASA used in ALMA Operations to calibrate and image the archived data will be the same as the publicly posted pipeline version available from the "Obtaining CASA" page, but they can be slightly different (e.g. for bugs that have an operational workaround but which are fixed in the posted public version). There is a "**Pipeline Version Tracker**" available from the Science Portal at http://almascience.org/processing/science-pipeline#version, which lists the versions of CASA+Pipeline used in ALMA Operations as well as the versions which should be used for any restoring or reprocessing of the data from the same cycle (as described elsewhere in this document).

3.4 Pipeline and CASA tasks

The pipeline heurisitics are written as special CASA tasks, where they appear with a `hif_` or `hifa_` (for interferometric) or `hsd_` (for single-dish) prefix. They can be viewed and executed within CASA in exactly the same way as other CASA tasks. For example, one can view the possible inputs for the task `hifa_importdata` by typing `inp hifa_importdata`. To see all the tasks available in CASA, type `tasklist`. The pipeline heuristics use CASA tasks wherever possible to perform the data reduction or imaging. E.g. the pipeline bandpass calibration task

`hifa_bandpass` calls the CASA `bandpass` tasks, and the interferometric imaging task `hif_makeimages` calls the CASA imaging task `tclean`.

The standard pipeline recipes are deterministic and should always give the same result for the same data. However, the CASA pipeline tasks are designed to be highly flexible, so that they can have the default inputs over-ridden with user-specified values, or be added, subtracted, or rearranged to produce alternative processing recipes. This enables a manual "mix and match" mode for data reduction and imaging that combines standard CASA pipeline tasks with other CASA commands or python code to produce scripts that are better tuned to the idiosyncrasies of a specific dataset. Some common "manual mode" modifications are presented in the Sec. 6 below. A complete list of the variables for each pipeline task is given in the ALMA Science Pipeline Reference Manual.

CASA pipeline tasks operate like other CASA tasks. In particular, the scope of variables follow CASA rules. This means that when CASA pipeline tasks are called with no arguments, they will assume any previously defined variables used by the task, whereas calling the same task with at least one argument will not. For example, typing the commands "`refant='DA45'; hifa_gfluxscale();`" will use the antenna named 'DA45' as the reference antenna, whereas typing "`refant='DA45'; hifa_gfluxscale(pipelinemode='interactive');`" will result in the pipeline picking a reference antenna according to its default heuristics.

This document, along with the Pipeline Reference Manual, describe key aspects of the CASA pipeline tasks. Important changes to other CASA tasks are documented in the Release Notes for the corresponding CASA release, available from the CASA page at https://casa.nrao.edu/current_casa.shtml.

4 Data Processing Files

4.1 Archived scripts

There are several scripts that are archived with ALMA data deliveries. These are fully described in the document **ALMA QA2 Data Products for Cycle 3** available from ALMA Science Portal under the "Processing" tab (or directly at https://almascience.nrao.edu/processing/qa2-data-products). The particular scripts for a specific dataset will also be described in the README file archived with the data products. This will vary based on how the data were processed (pipeline calibrated + imaged; pipeline calibrated & manually imaged; manually calibrated + pipeline imaged, manually calibrated + manually imaged).

The scripts produced by the pipeline are called **casa_pipescript.py** and, for data run through the IF calibration pipeline, **casa_piperestorescript.py**. The former includes all pipeline processing commands that were run on the data, and is more fully described below. The latter "restores" the data, which means that rather than re-running the pipeline calibration commands, it uses previously derived calibration and flagging tables and applies them directly to the raw data, producing a calibrated measurement set. This is much quicker and requires less computing resources than re-running the pipeline calibration commands. Currently, the restore capability is only available for data run through the IF pipeline; a SD pipeline calibrated dataset must be recreated by running the pipeline commands in the corresponding **casa_pipescript.py.**

Every delivery package includes a master script called **scriptForPI.py** that will reproduce the calibrated data regardless of how it was processed. This script is not created by the pipeline, but instead by the data packaging software so that it is produced for both pipeline and manually reduced data. For IF pipeline calibrated data, it will simply invoke the pipeline-produced **casa_piperestorescript.py** script. For SD pipelined data, it will invoke the corresponding

casa_pipescript.py. Instructions for using **scriptForPI.py** are included in the README file archived with each processed dataset, and in more detail in the document **ALMA QA2 Data Products for Cycle 3** referenced above.

Using scriptForPI.py is the recommended and fastest method of obtaining calibrated ALMA data from the delivery. However, to change the calibration results, one would re-run the commands in **casa_pipescript.py** after making modifications, as described in Sec. 5 below.

4.2 Pipeline "Helper" text files

Both the IF and SD pipeline use a number of text files that, if present, will affect the pipeline results (e.g. by applying manually identified flags or by updating calibrator fluxes or antenna positions before calculating the calibration tables). These files are particularly useful for users to over-ride the default pipeline behavior when re-running the pipeline at home, as more fully described in Sec. 6 below. They include the following:

- **flux.csv**: This file is used by the IF pipeline to update the flux of calibrators. The flux of the calibrator with the "AMPLITUDE" intent will affect the overall flux scale of the data. If this file is not present where the pipeline is run, the fluxes in the ASDM(s) will be used, representing the best flux estimate at the time the SB was executed. If no flux value appears in either the flux.csv file or the ASDM, a flux of 1.0 Jy is adopted.
- **jyperk.csv**: This file is used by the SD pipeline to set the "Jansky to Kelvin" calibration factors which set the overall fluxscale of the data. If it is not present where the pipeline is run, then a conversion factor of unity is assumed.
- **antennapos.csv**: This file is used by the IF pipeline to update the positions of the antenna elements. If it is not present where the pipeline is run, the positions in the ASDM(s) will be used.
- **uid*flagtemplate.txt**: This file is used to add additional CASA flagging commands that will be applied to the data before the calibration tables are calculated.
- **uid*flagtargetstemplate.txt**: This file is used to add additional CASA flagging commands that will be applied to the data after the calibration tables are calculated, but before science target imaging is performed.
- **cont.dat**: This file is used to specify the continuum frequency ranges used for constructing the continuum images and creating the continuum-subtracted cubes.

The format of each of these files is given in Sec. 6.

4.3 The Pipeline processing script: casa_pipescript.py

4.3.1 Format of casa_pipescript.py

The complete set of pipeline commands are given in the script **casa_pipescript.py**. This is a python script that includes all tasks and parameter values, in the correct sequence, that were used for the pipeline run. A typical **casa_pipescript.py** script for a SD Pipeline run (incuding both calibration+imaging steps) is shown in Figure 1, while a typical IF pipeline script that includes both pipeline calibration and imaging steps is shown in Figure 2.

For data that were both calibrated and imaged in the pipeline (including all SD data run through the pipeline), the **casa_pipescript.py** file will include both the calibration and imaging pipeline commands. For IF data that were calibrated in the pipeline but imaged outside of the pipeline, the **casa_pipescript.py** file will only include the IF calibration pipeline commands (up to the line "# Start of pipeline imaging commands" line in Figure 2), and the archived data will include a

separate **scriptForImaging.py** script containing the manual (CASA) imaging commands. If instead the IF data were manually calibrated and pipeline imaged, the **scriptForPI.py** would include the manual (CASA) calibration commands, and the IF pipeline imaging commands (those following the line "# Start of pipeline imaging commands" line in Figure 2) would be included in a separate **scriptForImaging.py** script.

```
__rethrow_casa_exceptions=True
h_init()

hsd_importdata(vis = ['uid___A002_X877e41_X452'])
hsd_flagdata(pipelinemode='automatic') ## Uses *flagtemplate.txt
hifa_tsyscal(pipelinemode='automatic')
hifa_tsysflag(fnm_byfield=True)
hsd_skycal(pipelinemode='automatic')
hsd_k2jycal(pipelinemode='automatic') ## Uses jyperk.csv
hsd_applycal(pipelinemode='automatic')
hsd_baseline(pipelinemode='automatic')
hsd_blflag(pipelinemode='automatic')
hsd_baseline(pipelinemode='automatic')
hsd_blflag(pipelinemode='automatic')
hsd_imaging(pipelinemode='automatic')

h_save()
```

Figure 1: Example of the Single Dish Pipeline calibration + imaging script casa_pipescript.py. The "##" comment line identifies the pipeline command that uses one of the pipeline "helper" text files described in Sec. 4.2.

```
__rethrow_casa_exceptions = True
h_init()
try:
    hifa_importdata(dbservice=False,
     vis=['uid___A002_X877e41_X452'], session=['session_1'])
                                          ## Uses flux.csv
    hifa_flagdata(pipelinemode="automatic")##Uses *flagtemplate.txt
    hifa_fluxcalflag(pipelinemode="automatic")
    hif_rawflagchans(pipelinemode="automatic")
    hif_refant(pipelinemode="automatic")
    hifa_tsyscal(pipelinemode="automatic")
    hifa_tsysflag(pipelinemode="automatic")
    hifa_antpos(pipelinemode="automatic") ## Uses antennapos.csv
    hifa_wvrgcalflag(pipelinemode="automatic")
    hif_lowgainflag(pipelinemode="automatic")
    hif_gainflag(pipelinemode="automatic")
    hif_setjy(pipelinemode="automatic")
    hifa_bandpass(pipelinemode="automatic")
    hifa_spwphaseup(pipelinemode="automatic")
    hifa_gfluxscale(pipelinemode="automatic")
    hifa_timegaincal(pipelinemode="automatic")
    hif_applycal(pipelinemode="automatic")
    hif_makeimlist(intent='PHASE,BANDPASS,CHECK')
    hif_makeimages(pipelinemode="automatic")
    hif_exportdata(pipelinemode="automatic")

# Start of pipeline imaging commands
    hif_mstransform(pipelinemode="automatic")
    hifa_flagtargets(pipelinemode="automatic")
                               ## Uses *flagtargetstemplate.txt
    hif_makeimlist(specmode='mfs')          ## Uses cont.dat
    hif_findcont(pipelinemode="automatic")  ## Modifies cont.dat
    hif_uvcontfit(pipelinemode="automatic") ## Uses cont.dat
    hif_uvcontsub(pipelinemode="automatic")
    hif_makeimages(pipelinemode="automatic")## Uses cont.dat
    hif_makeimlist(specmode='cont')         ## Uses cont.dat
    hif_makeimages(pipelinemode="automatic")## Uses cont.dat
    hif_makeimlist(width='')                ## Uses cont.dat
```

Figure 2: Example of an IF Pipeline casa_pipescript.py script for a dataset that was run through the Pipeline for both calibration and imaging. The "##" comment lines identify the pipeline commands that uses one of the pipeline "helper" text files described in Sec. 4.2.

.

The tasks names, order, and parameter values in the **casa_pipescript.py** script reflect the processing recipe used for each individual delivery. Additionally, the `pipelinemode` parameter is set to "automatic" for each task. In this mode, the task takes the default settings for each tasks and only a limited number of parameters are exposed for editing by a user. Setting the pipeline mode to "interactive" will usually enable the values of a larger number of parameters to be changed. To see the variables available in the pipeline "automatic" mode, type "`pipelinemode='automatic'; inp <task_name>`" at the CASA command line. To see the variables available in the pipeline "interactive" mode, type "`pipelinemode='interactive'; inp <task_name>`". See the **ALMA Science Pipeline Reference Manual** for more details, and Sec. 5 below for examples of modified pipeline re-runs.

4.3.2 Results from running the single dish casa_pipescript.py

Running the script will create:

- A calibrated, baseline subtracted MS for each ASDM with a name like `uid___A00X_XXXX_XXX.ms_bl`.
- Baseline subtracted image cubes of the the science targets in *.image format (1 per spectral window, at the native correlator frequency spacing).
- A **pipeline-*/html** directory containing
  - The Pipeline WebLog (see Sec. 9).
  - The **casa_commands.log** file (see Sec. 4.4).

4.3.3 Results from running the interferometric casa_pipescript.py

Running the script through the first `hif_makeimages` command (calibrator imaging) will create:

- A calibrated MS for each ASDM with a name like `uid___A00X_XXXX_XXX.ms`. This ms includes both calibrator and science data and all spectral windows, with the raw data in the DATA column, and the calibrated continuum + line data in the CORRECTED column.
- Continuum images of the bandpass, phase, and (if present) check source calibrators (1 per spectral window, in *.image format). To view a *.image file e.g. use casaviewer image_file_name.
- A **pipeline-*/html** directory containing:
  - The Pipeline WebLog (see Sec. 8).
  - The casa_commands.log file (see Sec. 4.4).

Running the script through `hif_mstransform` command will additionally create:

- A calibrated MS for each ASDM containing only science target data (only science targets and spectral windows), with a name like `uid___A00X_XXXX_XXX_target.ms`. This ms will have the raw data in the DATA column, and the calibrated continuum + line data in the CORRECTED column.

Running the script through `hif_uvcontsub` command will result in:

- The science-target only MS (`uid___A00X_XXXX_XXX_target.ms`), now with the calibrated continuum + line data in the DATA column, and the calibrated continuum subtracted data in the CORRECTED column.

Running the script through the final `hif_makeimages` command (science target spectral line imaging) will additionally create:

- Per-spw continuum images, aggregate continuum images, and continuum subtracted image cubes of at least some science targets (the number of targets may be reduced either automatically – see Sec. 8.20 – or manually).

4.4 CASA equivalent commands file: casa_commands.log

The **casa_commands.log** file is written by the pipeline to provide a list of the equivalent CASA task commands (as opposed to Pipeline tasks) used by the Pipeline to process a dataset. While this log cannot be used to create a CASA reduction script that is identical to the Pipeline processing, it provides executable CASA commands with the parameter settings used by the pipeline. The log is commented to indicate which Pipeline stage the tasks were called from and why. The imaging commands given in this file can be easily modified to produce new imaging products with more finely tuned inputs (e.g. interactive masks and deeper cleaning thresholds).

5 Modifying a Pipeline Run using casa_pipescript.py

5.1 Pipeline re-processing considerations

As a rule, it does not make sense to rerun the **casa_pipescript.py** exactly as delivered, since this will merely reproduce the calibrated measurement set (which for IF Pipeline calibrated data is much more easily generated using **scriptForPI.py** to "restore" the calibration, as described in Sec. 4.1 above) and/or already-delivered products. Instead, it is likely that the user may want to redo the calibration after some modifications or produced modified imaging products. This section describes a few of the more common calibration and imaging changes for both the IF and SD Pipeline tasks. See the **ALMA Science Pipeline Reference Manual** for more complete details on the pipeline tasks and their inputs.

Re-running the pipeline can be very resource-intensive, both from a compute-time and disk-space perspective. For the compute time, an idea of how long the pipeline took when can be inferred from the WebLog (using the **Execution Duration** shown on the top of the "Home" page of the WebLog – see Figure 9, or the **Task Execution Statistics** that are listed for each task in the "By Task" part of the WebLog – see e.g. Figure 13). Those times, however, reflect the run times using the ALMA Operations processing clusters, which have 64 – 256 GB RAM. Concerning disk space, to re-run SD or IF pipeline calibration, it is advisable to have a system with at least 8 GB RAM, and 50 – 75 GB free disk space per ASDM. To re-run the IF imaging pipeline, it is advisable to have a system with ≥64 GB RAM, and the available disk space needs to be 10 – 100 times the expected size of the final imaging products.

The above resource requirements for the IF imaging pipeline are rather daunting. However, In practice, it is unlikely that the imaging pipeline commands would need to be rerun in their entirety. It would be much quicker and demand much less computing resources to only image the sources and or spectral windows (spw) or channels of interest, at an appropriate spectral resolution. This can be done by finding the corresponding `tclean()` command in the provided **casa_commands.log** file, modifying it as desired, and running it in CASA. These commands work on the measurement set created by the pipeline `hif_mstransform()` command, so that part of the imaging script would need to be run first.

Please contact ALMA via the Helpdesk if assistance is needed with data reprocessing.

5.2 Preparing to run casa_pipescript.py

The following steps describe how to modify and re-run the Pipeline, starting from the products and directory structure created after downloading the data (see the **ALMA QA2 Data Products** document for details):

- Copy **casa_pipescript.py** from the **script** directory to the **raw** directory.
- To re-run IF calibration: copy **flux.csv, antennapos.csv** (if present), and **uid*flagtemplate.txt** from the **calibration** directory to the **raw** directory.
- To re-run IF imaging: copy **uid*flagtargetstemplate.txt** from the **calibration** directory to the **raw** directory. Copy **uid*cont.dat** from the **calibration** directory to **cont.dat** in the **raw** directory.
- To re-run SD calibration & imaging: copy **jyperk.csv** and **uid*flagtemplate.txt** from the **calibration** directory to the **raw** directory.

In the **raw** directory:

- Make sure the naming of the raw ALMA data is consistent with those provided in the script (e.g. if the data ends in .**asdm.sdm** then move to names which do not have this suffix).
- Modify the pipeline "helper" files as desired (e.g. editing the ***flagtemplate.txt** file to add any additional flags – see Sec. 6 for other options).
- Edit **casa_pipescript.py** to only include the pipeline steps you wish to repeat (e.g. commenting out the findcont or imaging steps, which are very computationally expensive).
- Start the version of CASA containing Pipeline using `casapy --pipeline`

You are now ready to run the script by typing `execfile('casa_pipescript.py')`. Alternatively, you can sequentially execute individual commands from **casa_pipescript.py**, stopping at any point to run other CASA commands (plotms, etc).

Note that to re-run the Pipeline multiple times, it is recommended to start each time from a clean directory containing only the raw data, CASA "helper" text files, and the casa_pipescript.py script.

5.3 Modifying Calibration Commands

The pipeline calibration commands can be modified to produce different results.

For instance, problematic datasets (ASDMs) can be excluded from the processing by editing the "`vis=`" and "`session=`" lists in `hifa_importdata` or `hsd_importdata` tasks in the **casa_pipescript.py** script.

As a second example, a user-specified prioritized reference antenna list can be specified via the "refant" variable in calibration tasks, over-riding the pipeline reference antenna heuristics, by switching to pipelinemode='interactive' and passing the desired refant list. E.g.

```
hifa_bandpass(pipelinemode="interactive", refant='DV06,DV07')
```

See the Pipeline Reference Manual for more options.

Another use case is to keep the default pipeline commands, but to changes the values in the Pipeline "helper" text files to e.g. change the flux scaling, or update antenna positions (see Sec. 6 for details). The new values will be used when **casa_pipescript.py** is executed.

5.4 Modifying IF Pipeline Imaging Commands

The pipeline imaging commands can be modified to produce different products. Typical reasons for re-imaging include:

- Imaging improvements to be gained from interactively generating emission specific clean mask and cleaning more deeply. The Cycle 4 pipeline uses a generic clean mask, and a conservative clean threshold (see Sec. 8.28 for specifics). Cases with moderate to strong emission (or absorption) can significantly benefit from deeper clean with interactive clean masking, with the most affected property being the integrated flux density. For S/N greater than about 100, the images can also often be improved by self-calibration coupled with deeper clean with manual clean masks.
- Non-optimal continuum ranges. The pipeline uses heuristics that attempt to identify continuum channels over a very broad range of science target line properties. Particularly for strong line forests (hot-cores) and occasionally for TDM continuum projects the pipeline ranges can be non-optimal -- too much in the first case and too little in the second.

Other science goal driven reprocessing needs may include:

- Desire to bin channels in the imaging stage to increase the S/N of cubes.
- Desire to use a different Briggs Robust image weighting than the default of robust=0.5 (smaller robust = smaller beam, poorer S/N; larger robust = larger beam, better S/N).
- Desire to uv-taper images to to increase the S/N for extended emission.
- Desire to use different continuum frequency ranges than determined by the pipeline, by modifying the cont.dat file (Sec. 6.6).

Some re-imaging examples are given in a "CASA Guide" at https://casaguides.nrao.edu/index.php/ALMA_Imaging_Pipeline_Reprocessing. There you will find examples of the following:

- Making aggregate continuum image with all channels of all spectral windows.
- Redoing continuum subtractions with user-derived continuum ranges.
- Making a cube of subset of sources, spectral windows, with a different robust weight and channel binning factor.

5.5 Manual imaging after running casa_pipescript.py

5.5.1 SD Data

After calibration with the script **casa_pipescript.py**, it is possible to re-image using the CASA Single Dish task, `sdimaging`, with user-defined parameters. As mentioned earlier, the Single Dish Pipeline creates a calibrated MS with a filename extension of "***.ms_bl**" for each ASDM. The `sdimaging` command will make images of all MS that are specified in the **infiles** parameter. For other parameters in `sdimaging`, refer to the ***casa_commands.log** file.

Note that the images included in the delivery package have the native frequency resolution and a cell size of one-ninth of the beam size, as recommended in the SD "CASA Guide" (https://casaguides.nrao.edu/index.php/M100_Band3_SingleDish_4.5). If you want to change the frequency resolution and cell size, we recommend that you import the delivered FITS data cubes into CASA and regrid them using the CASA task `imregrid`.

It is also possible to revise the baseline subtraction using your prefered mask range instead of the pipeline-defined range. We recommend doing this on the images using the CASA tasks `imcontsub` or `tsdbaseline` during your own manual calibration (refer to the CASA Guides).

5.5.2 IF Data

For IF data that are pipeline calibrated but manually imaged, the imaging commands will be included in a separate **scriptForImaging.py** script, containing all the CASA commands used to create the delivered products. In order to use this imaging script after using **casa_pipescript.py** or **scriptForPI.py** to recalibrate, the science spectral windows must first be "split" out from the calibrated measurement sets and the measurement sets output with a `.split.cal` suffix. To perform the split, in CASA e.g.:

```
split('uid__A002_X89252c_X852.ms',
    outputvis='uid__A002_X89252c_X852.ms.split.cal', spw='17,19,21,23')
```

The science spectral windows are specified in the Pipeline WebLog (Home > Observation Summary > Measurement Set Name > Spectral Setup, in the ID column) or can be determined using the CASA task listobs e.g. `listobs('uid___A002_X89252c_X852.ms')`, where the results will be reported in the CASA logger.

If a script named **scriptForFluxCalibration.py** is present in the script directory, this must also be executed prior to running **scriptForImaging.py**.

5.6 Manipulating the Pipeline Context

It is recommended to always run the Pipeline using python scripts. New Pipeline runs/scripts need to be initialised using `h_init` in order to create an empty pipeline **Context**.

If the script is modified to only run a subset of the pipeline tasks, the **Context** should be saved after the last task by using `h_save`. To resume the run, use `h_resume` to load the saved **Context** before executing any pipeline tasks. See the **ALMA Science Pipeline Reference Manual** for more information.

To use the Pipeline to calibrate a dataset but to e.g. insert a different bandpass table into the processing, the following procedure should be followed:

- Run the pipeline until the end of the bandpass table creation task `hifa_bandpass`.
- View the calibration tables that Pipeline will use with `hif_show_calstate`.
- Export the calibration tables Pipeline uses to a file on disk using `hif_export_calstate`.
- Edit the calstate file to replace the name of the Pipeline-created bandpass table with the one it is wanted to use instead.
- Import the edited calstate file back to the **Context** using `hif_import_calstate` and resume the processing.

6 Description of Pipeline "Helper" Text Files

As mentioned in Sec. 4.2, both the IF and SD pipeline use a number of text files that are read by various pipeline tasks (as indicated in Figure 1 & 2), and which affect the pipeline results (e.g. by applying manually identified flags or by updating calibrator fluxes or antenna positions before calculating the calibration tables). These files are particularly useful for users to over-ride the default pipeline behavior when re-running the pipeline at home, as described in the following section. Below we describe all of the currently available control files, identifying whether they are used by the IF pipeline, SD pipeline, or both in the subsection heading.

6.1 IF Pipeline: flux.csv

From Cycle 4 onward, the fluxes of standard ALMA quasar calibrators at the observed frequencies for each spw are written into the ASDM, using extrapolated values calculated from entries in the ALMA Source Catalog available at the time of execution. These fluxes are sometimes updated subequently (thereby bracketing the observation in time), allowing for more accurate interpolated fluxes to be used for the absolute flux calibration.

Since the pipeline is usually run days to weeks after an observation, ALMA staff run commands outside of the pipeline to get the best-available fluxes for standard calibrators at the time the pipeline is run (in a future release, this query will be done automatically by the pipeline). These are written into the **flux.csv** text file, which is then read in by the pipeline `hifa_importdata` task (if it exists in the directory where the pipeline is run) and used to over-ride the values in the ASDM. The new flux value of the flux calibrator (the source with intent=AMPLITUDE) is then used in the subsequent `hif_setjy` task. Values for the other calibrator intents (BANDPASS, PHASE, CHECK) are also updated, but these values are only shown for comparison against the values derived from the pipeline calibration calibration (both are shown in a table in the `hifa_gfluxscale` stage of the WebLog – see Sec. 8.15). If the **flux.csv** file is not available where the pipeline is run, then the values entered into the ASDM by the online system are used. If **flux.csv** file is not available where the pipeline is run and there are no values entered by the online system, then a flux of 1 Jy will be assumed.

The format of the **flux.csv** file is shown in Figure 3 below. It contains one row for every spw of every calibrator (intents of AMPLITUDE, BANDPASS, PHASE or CHECK) in every ASDM in the MOUS. This file can be edited by users and the pipeline re-run in order to scale the fluxes of each ASDMs to a different value for the AMPLITUDE calibrator. Changing the values of other calibrators will not have an effect on the calibration.

```
Ms,field,spw,I,Q,U,V,spix,comment
uid___A002_Xbb63ba_X1626.ms,0,19,0.9507,0.0,0.0,0.0,-0.308650436472,"J0519-4546
intent=AMPLITUDE,ATMOSPHERE,BANDPASS,POINTING,WVR # +-0.1252Jy, fre
q=226.113GHz, spec_index=-0.309+-0.031, Band3/7_separation=0 days, meanAge=1 days, setjy parameters for field 0 (J0519-4546):
spix=-0.3087, reffreq='226.1127GHz', fluxdensity=[0.950651,0,0,0], au.getALMAFluxcsv v1.3454 executed on 2017-02-02 19:05:02 UT"
uid___A002_Xbb63ba_X1626.ms,0,21,0.9527,0.0,0.0,0.0,-0.308650436472,"J0519-4546
intent=AMPLITUDE,ATMOSPHERE,BANDPASS,POINTING,WVR # +-0.1242Jy, freq=224.561GHz"
uid___A002_Xbb63ba_X1626.ms,1,19,0.580241916331,0.0,0.0,0.0,0.0,"J0550-5732 intent=ATMOSPHERE,PHASE,WVR"
uid___A002_Xbb63ba_X1626.ms,1,21,0.583046038078,0.0,0.0,0.0,0.0,"J0550-5732 intent=ATMOSPHERE,PHASE,WVR"
uid___A002_Xbb63ba_X18b0.ms,0,19,0.897297453643,0.0,0.0,0.0,0.0,"J0519-4546 intent=AMPLITUDE,ATMOSPHERE,BANDPASS,POINTING,WVR"
uid___A002_Xbb63ba_X18b0.ms,0,21,0.899465684438,0.0,0.0,0.0,0.0,"J0519-4546 intent=AMPLITUDE,ATMOSPHERE,BANDPASS,POINTING,WVR"
uid___A002_Xbb63ba_X18b0.ms,1,19,0.580241986645,0.0,0.0,0.0,0.0,"J0550-5732 intent=ATMOSPHERE,PHASE,WVR"
uid___A002_Xbb63ba_X18b0.ms,1,21,0.583046108732,0.0,0.0,0.0,0.0,"J0550-5732 intent=ATMOSPHERE,PHASE,WVR"
```

Figure 3: Example of a flux.csv file used by the interferometric pipeline (one per MOUS)

6.2 SD Pipeline: jyperk.csv

ALMA single-dish observations do not include observations of absolute amplitude calibrators. Instead, the observatory conducts regular observations of standard single-dish calibrators and stores them in an observatory database. When the single-dish pipeline is run, ALMA staff run commands outside the pipeline to extract the best value of these “Jansky to Kelvin” calibration factors, based on the observing date, frequency, Tsys, and source elevation. The appropriate values are written into a the **jyperk.csv** text file that is read and applied when the `hsd_k2jycal` task is run.

The format of the **jyperk.csv** file is shown in Figure 4 below. It contains one row for every spw in every ASDM in the MOUS. This file can be edited by users and the pipeline re-run in order to scale the fluxes of each ASDM to a different value.

```
MS,Antenna,Spwid,Polarization,Factor
uid___A002_Xb1d975_Xf65.ms,PM02,17,I,43.785
uid___A002_Xb1d975_Xf65.ms,PM02,19,I,43.782
uid___A002_Xb1d975_Xf65.ms,PM02,21,I,43.664
uid___A002_Xb1d975_Xf65.ms,PM02,23,I,43.63
uid___A002_Xb1d975_Xf65.ms,PM02,25,I,43.638
uid___A002_Xb1d975_Xf65.ms,PM02,27,I,43.64
uid___A002_Xb1d975_Xf65.ms,PM02,29,I,43.641
uid___A002_Xb1d975_Xf65.ms,PM04,17,I,43.785
uid___A002_Xb1d975_Xf65.ms,PM04,19,I,43.782
uid___A002_Xb1d975_Xf65.ms,PM04,21,I,43.664
uid___A002_Xb1d975_Xf65.ms,PM04,23,I,43.63
uid___A002_Xb1d975_Xf65.ms,PM04,25,I,43.638
uid___A002_Xb1d975_Xf65.ms,PM04,27,I,43.64
uid___A002_Xb1d975_Xf65.ms,PM04,29,I,43.641
uid___A002_Xb1cc39_X1e46.ms,PM02,17,I,43.782
uid___A002_Xb1cc39_X1e46.ms,PM02,19,I,43.778
uid___A002_Xb1cc39_X1e46.ms,PM02,21,I,43.661
uid___A002_Xb1cc39_X1e46.ms,PM02,23,I,43.627
uid___A002_Xb1cc39_X1e46.ms,PM02,25,I,43.635
uid___A002_Xb1cc39_X1e46.ms,PM02,27,I,43.636
uid___A002_Xb1cc39_X1e46.ms,PM02,29,I,43.638
uid___A002_Xb1cc39_X1e46.ms,PM04,17,I,43.782
uid___A002_Xb1cc39_X1e46.ms,PM04,19,I,43.778
uid___A002_Xb1cc39_X1e46.ms,PM04,21,I,43.661
uid___A002_Xb1cc39_X1e46.ms,PM04,23,I,43.627
uid___A002_Xb1cc39_X1e46.ms,PM04,25,I,43.635
uid___A002_Xb1cc39_X1e46.ms,PM04,27,I,43.636
uid___A002_Xb1cc39_X1e46.ms,PM04,29,I,43.638
```

Figure 4: Example of a jyperk.csv file used by the single-dish pipeline (one per MOUS)

6.3 IF Pipeline: antennapos.csv

The position of every antenna in an interfermetric observation must be known in order to properly transfer the calibration from the phase calibrator to the science targets. If these positions have errors, it will lead to phase errors in the imaging of the science target (increasing with telescope position error and separation between the phase calibrator and science target).

The antenna positions are calculated by special observatory observations taken outside of PI science observing, and the positions stored in an observatory database. This database is queried at the time at the time of an SB execution, and the approriate antenna positions are written into the ASDM. These positions are sometimes updated subequently, especially if the observation happened close to an array reconfiguration or if an array element was recently moved.

Since the pipeline is usually run days to weeks after an observation, ALMA staff run commands outside of the pipeline to get the best-available antenna positions at the time the pipeline is run (in a future release, this query will be done automatically by the pipeline). These are written into the **antennapos.csv** text file, which is then read in by the pipeline `hifa_antpos` task (if it exists in the directory where the pipeline is run) and used to over-ride the values in the ASDM.

The format of the **antennapos.csv** file is shown in Figure 5 below. It contains one row for every antennae every ASDM in the MOUS. This file can be edited by users and the pipeline re-run in order to correct antenna position errors.

```
name,antenna,xoff,yoff,zoff,comment
uid___A002_Xbb63ba_X18b0.ms,DA42,1.27536e-04,-3.54105e-04,-2.38014e-04,
uid___A002_Xbb63ba_X18b0.ms,DA46,1.98098e-04,-5.34528e-04,-3.65393e-04,
uid___A002_Xbb63ba_X18b0.ms,DA49,1.69321e-04,-2.81896e-04,-1.76309e-04,
uid___A002_Xbb63ba_X18b0.ms,DA52,-4.06882e-05,3.45109e-04,3.15047e-04,
uid___A002_Xbb63ba_X18b0.ms,DA62,-1.79249e-04,2.50696e-04,7.12701e-05,
uid___A002_Xbb63ba_X18b0.ms,DV03,-3.92453e-04,2.85912e-04,3.14499e-04,
uid___A002_Xbb63ba_X18b0.ms,DV08,-2.76083e-04,7.41071e-04,1.87197e-04,
uid___A002_Xbb63ba_X18b0.ms,DV14,-5.41156e-05,2.61746e-04,3.44329e-04,
uid___A002_Xbb63ba_X18b0.ms,DV15,-1.21313e-04,3.67910e-04,1.49062e-04,
uid___A002_Xbb63ba_X18b0.ms,DV23,1.73257e-04,1.36402e-04,-1.23099e-04,
uid___A002_Xbb63ba_X18b0.ms,DV25,3.12879e-03,-5.08802e-03,-2.87630e-03,
uid___A002_Xbb63ba_X18b0.ms,PM03,1.78948e-04,-5.00918e-04,-2.14580e-04,
uid___A002_Xbb63ba_X1626.ms,DA42,1.27536e-04,-3.54105e-04,-2.38014e-04,
uid___A002_Xbb63ba_X1626.ms,DA46,1.98098e-04,-5.34528e-04,-3.65393e-04,
uid___A002_Xbb63ba_X1626.ms,DA49,1.69321e-04,-2.81896e-04,-1.76309e-04,
uid___A002_Xbb63ba_X1626.ms,DA52,-4.06882e-05,3.45109e-04,3.15047e-04,
...
```

Figure 5: Example of a antennapos.csv file used by the interferometric pipeline (one per MOUS)

6.4 Both IF & SD Pipeline: uid*flagtemplate.txt

The pipeline flagging heuristics may prove inadequate, and users may wish to add additional flagging commands to exclude these data from the calibration. These manually-identified flags can be introduced to any Pipeline reduction by editing the **uid*flagtemplate.txt** files that are provided with the archived pipeline products and rerunning the pipeline calibration steps. There should be one file for every MS that needs additional flagging, with a name matching the MS uid. The flag commands can be any valid CASA `flagdata` command. For interferometic data, use the <AntID> syntax to flag only cross-correlation data for <AntID>, while for single dish data use the "<AntID>&&*" syntax to flag both cross- and auto-correlation data for <AntID>, and the "<AntID>&&&" syntax to flag auto-correlation data for <AntID>. Examples of the syntax to use in editing these files are given at the top of the files **uid*flagtemplate.txt (**see Figure 6).

These flag files will be picked up by the `hifa_flagdata/hsd_flagdata` tasks which are run before the calibration tasks, therefore excluding the manually identified data from being used to generate the calibration tables.

```
# User flagging commands file for the calibration pipeline
#
# Examples
# Note: Do not put spaces inside the reason string !
#
# mode='manual' correlation='YY' antenna='DV01;DV08;DA43;DA48&DV23' spw='21:1920~2880' autocorr=False
reason='bad_channels'
# mode='manual' spw='25:0~3;122~127' reason='stage8_2'
# mode='manual' antenna='DV07' timerange='2013/01/31/08:09:55.248~2013/01/31/08:10:01.296' reason='quack'
#
mode='manual' timerange='2016/12/05/03:55:30.1440' reason='applycal_outlier_amp'
mode='manual' antenna='PM02&&&' reason='PRTSIR2995'
```

Figure 6: Example of a uid*flagtemplate.text file used by both the interferometric and single-dish pipeline (one per ASDM)

6.5 IF Imaging Pipeline: uid*flagtargetstemplate.txt

Currently, there are no science target specific flagging heuristics in the IF pipeline, so errant data may be present, affecting the science imaging products. Users should examine the science data (e.g. using the CASA task `plotms`, or examing at the MS using the CASA viewer). If bad

data are found, flagging commands can be added to the **uid*flagtargetstemplate.txt** files that are provided with the archived pipeline products to exclude these data from subsequent imaging. There should be one file for every MS that needs additional flagging, with a name matching the MS uid. As for the **uid*flagtemplate.txt** files, the flag commands can be any valid CASA `flagdata` command. If these files are found in the directory where the pipeline is run, they will be picked up by the `hifa_flagtargets` task and applied to the data before science target imaging.

6.6 IF Imaging Pipeline: cont.dat

The pipeline-identified continuum frequency ranges, in LSRK units, for each spectral window of each source are entered into a file called **cont.dat** that is delivered with the pipeline products. This file lists the LSRK frequency ranges that were used to make the per-spw and aggregate continuum images, and for fitting and subtracting the continuum for the image cubes. When this file is in the directory where the pipeline is (re)run, the pipeline will use these entries directly instead of using its own heuristics (via the `hif_findcont` task) to determine them. Therefore, a user can edit this file (or create their own) in order to use a different continuum range. Alternatively, a user-defined file name can be passed as an argument to the `hif_makeimlist` task. An example **cont.dat** file is shown in Figure 7.

```
Field: G09_0850-0019
SpectralWindow: 17
NONE

SpectralWindow: 19
337.659971874~339.253995016GHz LSRK

SpectralWindow: 21

SpectralWindow: 25
349.755169752~351.067897111GHz LSRK
351.271057297~351.380451244GHz LSRK
```

Figure 7: Example of a cont.dat file used by the interferometric pipeline (one per MOUS). This example is for an MOUS that has 5 spectral windows; the entry for spw 21 is empty and spw 23 is omitted, which will result in the hif_findcont command determining the frequency ranges for these spectral windows.

The behavior of `hif_findcont` and the subsequent continuum subtraction and continuum and line imaging commands is as follows:

- If the spw line is followed by one or more frequency ranges, `hif_findcont` will not run its heuristics on the spw. The task `hif_uvcontfit` will use these frequency ranges to fit and subtract the continuum from this spw. Subsequent continuum images will include only these frequency ranges for this spw, and the spw line cubes will be made from the continuum subtracted data.
- If the spw line is followed by a line containing “NONE”, `hif_findcont` will not run its heuristics on the spw (if the delivered **cont.dat** file contains spw entries with “NONE”, this indicates that the `hif_findcont` task failed to find any continuum frequency ranges). The task `hif_uvcontfit` will skip fitting this spw. Subsequent continuum images will include the full frequency range for this spw (logging a warning), and the spw line cubes will have had no continuum subtraction performed.

- If an spw is not followed by a frequency range or is missing from **cont.dat** when `hif_findcont` is run, then it will try to find the frequency ranges, and these will be used to make subsequent continuum images, and for continuum subtraction.
- If an spw is not followed by a frequency range is missing from **cont.dat** when `hif_uvcontfit` is run, it will skip fitting this spw. Subsequent continuum images will include the full frequency range for this spw (logging a warning), and the spw line cubes will have had no continuum subtraction performed.

7 The Pipeline WebLog

This section gives overview of the Pipeline WebLog, which is a collection of webpages with diagnostic messages, tables, figures, and "Quality Assurance" (**QA**) scores. It is reviewed, long with the pipeline calibration and imaging products, as part of the ALMA Quality Assurance process, but also provides important information to investigators on how the pipeline calibration and imaging steps went.

The section describes common elements to the single dish and interferometic Pipeline WebLogs. Subsequent sections present descriptions of the SD- or IF- specific "By Task" part of the WebLog.

7.1 Overview

The WebLog is a set of html pages that give a summary of how the calibration of ALMA data proceeded, of the imaging products, and provides diagnostic plots and Quality Assurance (QA) scores. The WebLog will be in the **qa** directory of an ALMA delivery. To view the WebLog, untar and unzip the file using e.g. **tar zxvf *WebLog.tar.gz .** This will provide a **pipeline*/html** directory containing the WebLog, which can be viewed using a web browser e.g. **firefox index.html**.

The WebLog provides both a quick overview of datasets and also gives methods for exploring each pipeline stage in detail. Therefore most calibration pages of the WebLog will first give a single "representative" view, with further links to a more detailed view of all the plots associated with that calibration step. Some of these will have a "Plot command" link that provides the CASA command to reproduce the plot (see Figure 8). For some stages, the detailed plots can be filtered by a combination of outlier, antenna and spectral window criteria. Where histograms are displayed, in modern web browsers it is possible to draw boxes on multiple histograms to select the plots associated with those data points. All pipeline stages are assigned QA score to give an "at a glance" indication of any trouble points.

WebLog Quick Tips

- Any text written in blue, including headings, is a link to further information.
- To go straight to viewing calibrated science target plots, go to **By Task > hif_applycal** and scroll down to the bottom.
- Histograms can have selector boxes drawn on them using the mouse.
- The CASA commands for re-creating many of the WebLog plots are provided.

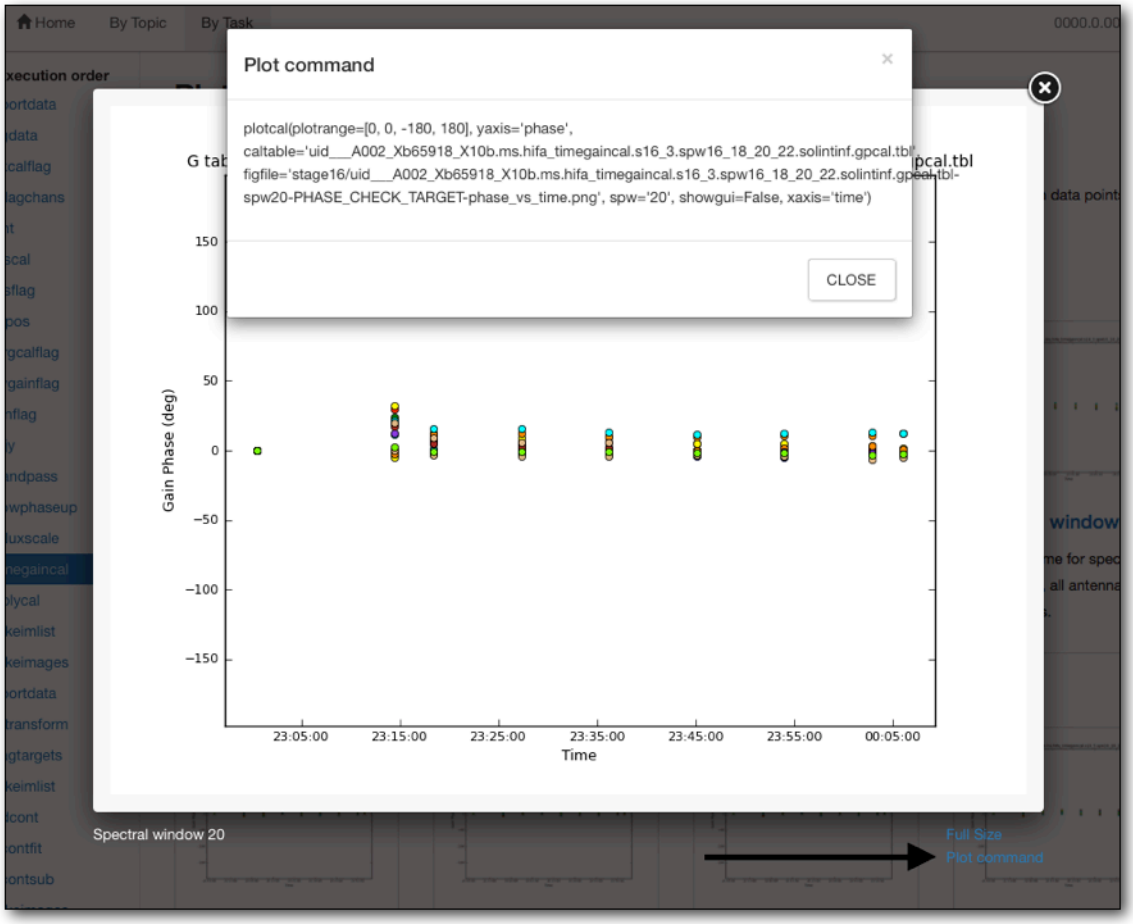


Figure 8: Example of WebLog plot with a "Plot command" link (arrow) that provides the CASA command for reproducing the plot.

7.2 Navigation

To navigate the main pages of the WebLog, click on items given in the bar at the top of the WebLog home page. Also use the **Back** button provided at the upper right on some of the WebLog sub-pages. Avoid using “back/previous page” on your web browser (although this can work on modern browsers). Throughout the WebLog, links are denoted by text written in blue and it is usually possible to click on thumbnail plots to enlarge them.

7.3 Home Page

The first page in the WebLog gives an overview of the observations (proposal code, data codes, PI, observation start and end time), a pipeline execution summary (pipeline & CASA versions, link to the current pipeline documentation, pipeline run date and duration), and an **Observation Summary** table. Clicking on the bar at the top of the home page (see Figure 9) enables navigation to **By Topic** or **By Task**.

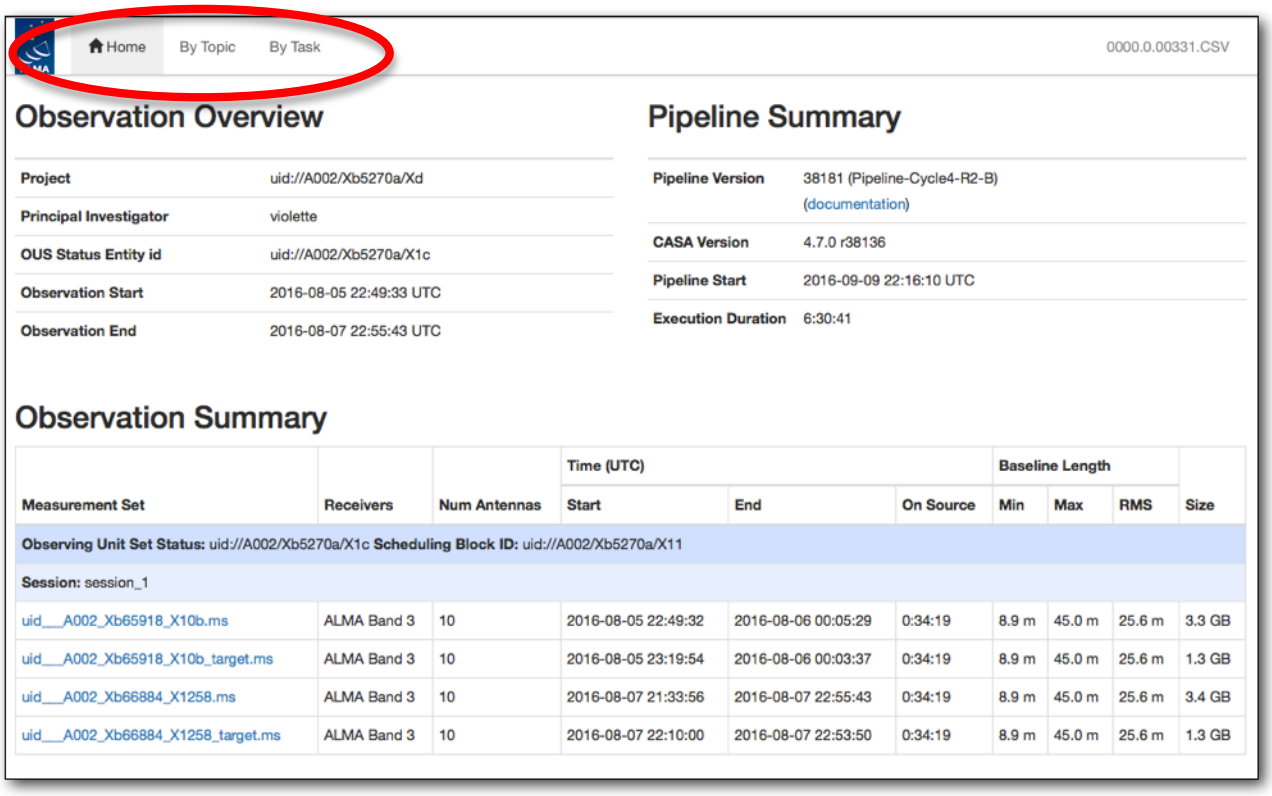


Figure 9: WebLog Home Page. The Navigation Bar is circled in red.

The **Observation Summary** table lists all the measurement sets included in the pipeline processing, grouped by observing "sessions". Each measurement set is calibrated independently by the pipeline. For data that have been run through the imaging stages of the pipeline, two MS will be listed – the original one including all data and spectral windows, and a target.ms containing only science target data. The table provides a quick overview of the ALMA receiver band used, the number of antennas, the start/end date and time, the time spent on source, the array minimum and maximum baseline length, the rms baseline length and the size of that measurement set. To view the observational setup of each measurement set in more detail, click on the name of it to go to its overview page.

7.3.1 Measurement set Overview pages

Clicking on the measurement set name in the **Observational Summary** table brings up the **Measurement set Overview page** (Figure 10). Each measurement set **Overview** page has a number of tables: **Observation Execution Time**, **Spatial Setup (includes mosaic pointings)**, **Antenna Setup**, **Spectral Setup** and **Sky Setup (includes elevation vs. time plot)**. For more information on the tables titled in blue text, click on these links. There are additionally links to **Weather, PWV, Scans** and **Telescope Pointings** (in the case of Single Dish observations) information. Two thumbnail plots, which can be enlarged by clicking on them, show the observation structure either as **Field Source Intent vs Time** or **Field Source ID vs Time**. To view the CASA listobs output from the observation, click on **Listobs Output**.

Figure 10: Measurement Set Overview Page. Click on the table headings in blue for more information about each.

7.4 By Topic Summary Page

The **By Topic** summary page provides an overview of all **Warnings and Errors** triggered, a Quality Assessment overview in **Tasks by Topic** and **Flagging Summaries** for the processing.

7.5 By Task Summary Page

The **By Task** summary page (Figure 11) gives a list of all the pipeline stages performed on the dataset. It is not displayed per measurement set as the Pipeline performs each step on every measurement set sequentially before proceeding to the next step; e.g. it will import and register all measurement sets with the Pipeline before proceeding to perform the ALMA deterministic flagging step on each measurement set. The name of each step on the By Task page is a link to more information. On the right hand side of the page are colored bars and scores that indicate how well the Pipeline processing of that stage went. Green bars should indicate a fairly problem-free dataset, while blue or red bars indicate less than perfect QA scores. Encircled symbols to the left of each task name ("?", "!" or "x"), indicate that there are informative QA messages on the subtask pages.

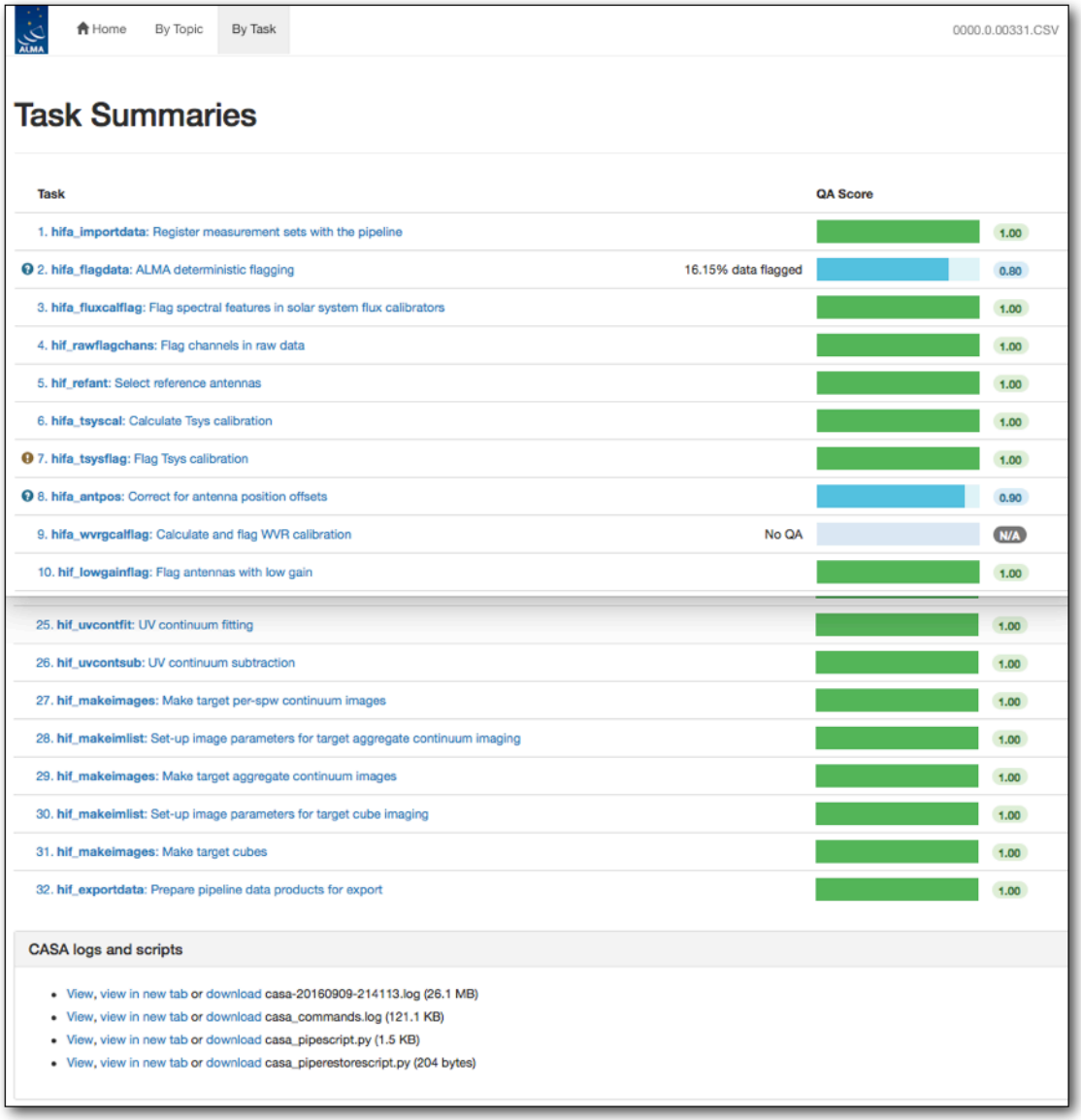


Figure 11: By Task summary view. The page has been truncated so both the top and bottom can be seen. Each pipeline stage is listed, along with its QA score (colored bars to the right), and links to the CASA logs and scripts.

7.5.1 CASA logs and scripts

At the bottom of the **By Task** summary page are links to the CASA logs and supporting files and scripts. These include the complete CASA log file produced during the pipeline run, the pipeline restoration scripts described in Sec. 4.1: **casa_pipescript.py** and **casa_piperestorescript.py**, and the **casa_commands.log** file described in Sec. 4.4.

7.6 Task Pages

Each task has its own summary page that is accessed by clicking on the task name on the **By Task** summary page or in the left navigation menu from other pages. The task pages provide the outcome, or the representative outcome, of each Pipeline task executed. **For a fast assessment of the calibration results, go straight to the applycal page.** At the top of the page will be any Task Notification (see Figure 12). These provide informative messages or warnings generated from the QA scoring and should be reviewed carefully.

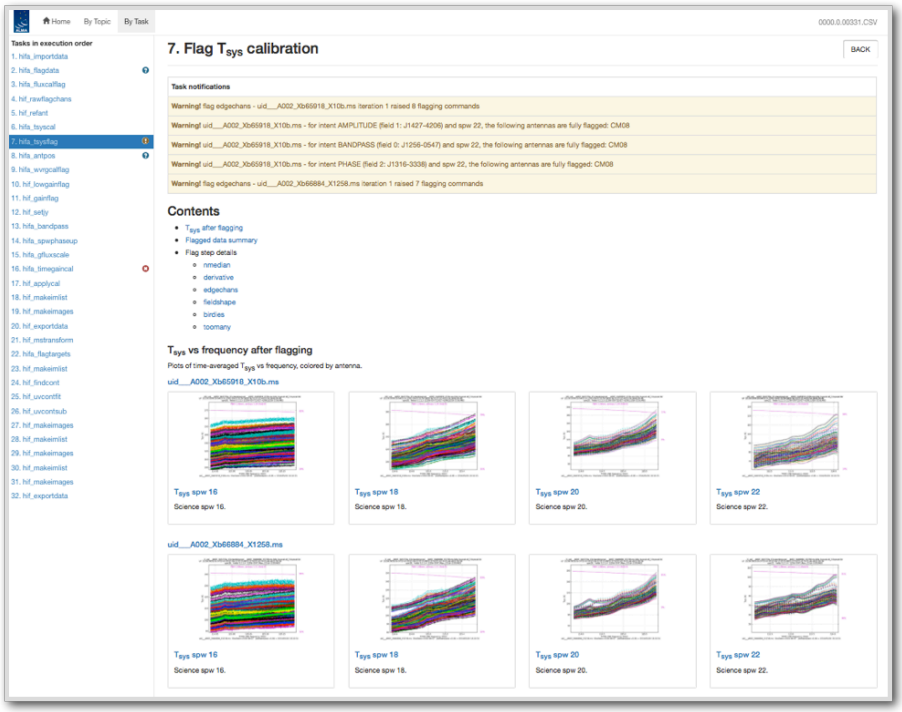


Figure 12: hifa_tsysflag task page, showing the task notifications at the top, and diagnostic plots (Tsys for each spw grouped by MS). Further down on the page are flagging summary tables. To see the sub-page for this task, click on the measurement set name in blue above each set of plots. This will take you to a page of detailed plots for individual MS/antenna/spectral windows (see Figure 14 for an example).

At the bottom of each task page are expandable sections for **Pipeline QA**, **Input Parameters** and **Task Execution Statistics**, and links to the CASA log commands for the specific task. An example is given in Figure 13.

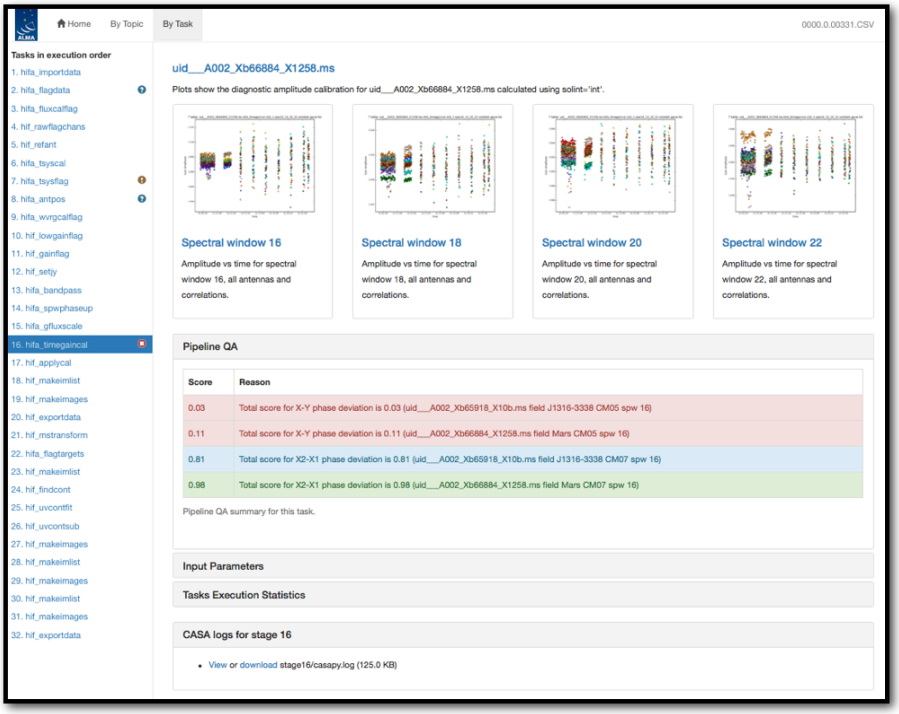


Figure 13: Bottom of the hifa_timegaincal page, showing the expanded Pipeline QA section, as well as the expandable sections for Input Parameters, Task Execution Statistics and link to the CASA logs for this stage.

7.6.1 Task sub-pages and plot filtering

Most sub-pages have further links in order to access a more detailed view of the outcome of each task. These links are often labelled by the measurement set name. Some of these plots can be filtered by entering one or more MS, antenna, or spectral window in the appropriate box. Still others have histograms of various metrics than can be selected using the cursor in a drop-and-drag sense to outline a range of histogram values and displays the plots for the MS/antenna/spw combinations that are responsible for those histogram values. An example of these subpages and plot filtering is given in Figure 14 – Figure 16 below, using the **By Task > hifa_tsysflag: Flag Tsys calibration** pages.

Figure 14: Unfiltered view of the hifa_tsysflag sub-page. The page is arrived at by clicking on the measurement set link from the hifa_tsysflag task page (Figure 12). Only the first row of plots are shown; many more appear below (one for each MS, antenna, spw combination). This page has histograms of three metric scores based on the median Tsys that can also be used to filter the plots that are displayed.

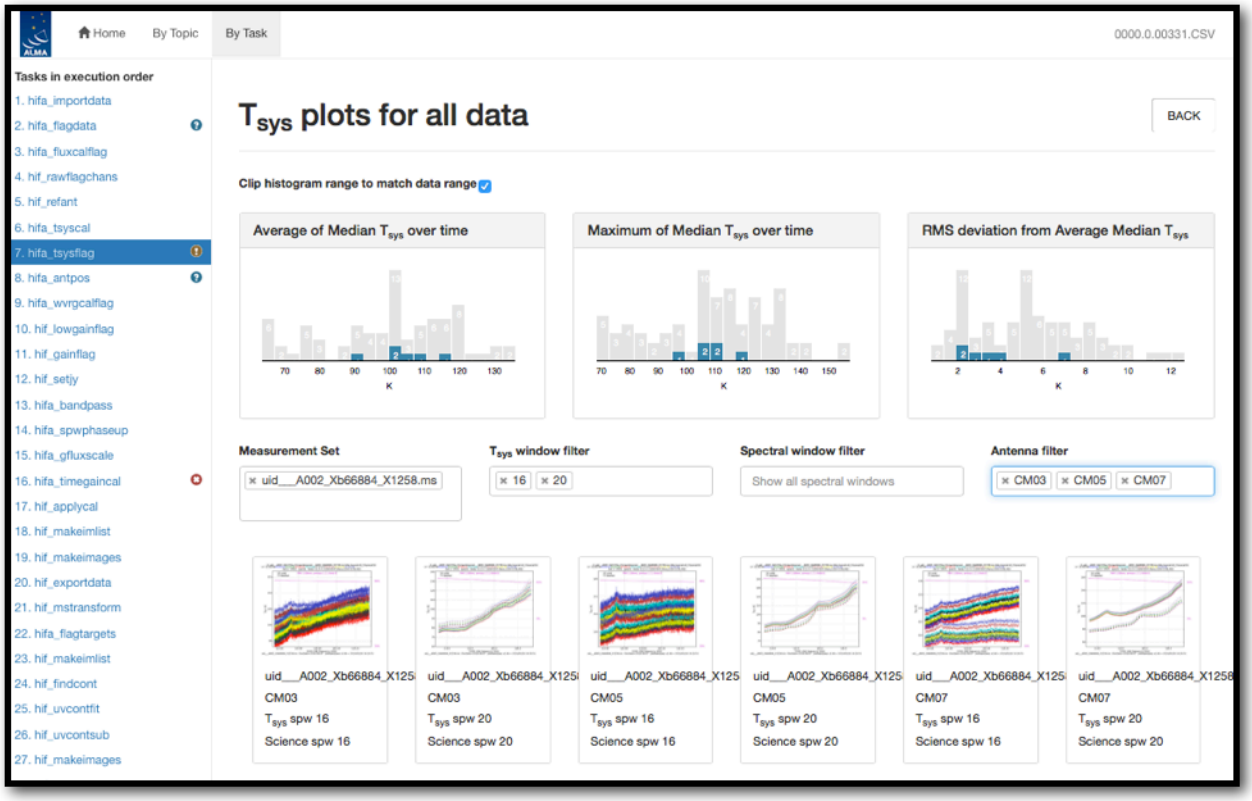


Figure 15: Same as Figure 14, but with a specific MS, Tsys window, and antenna filter set. The corresponding plots are displayed below, and their metric scores are shown by blue shading in the histogram plots.

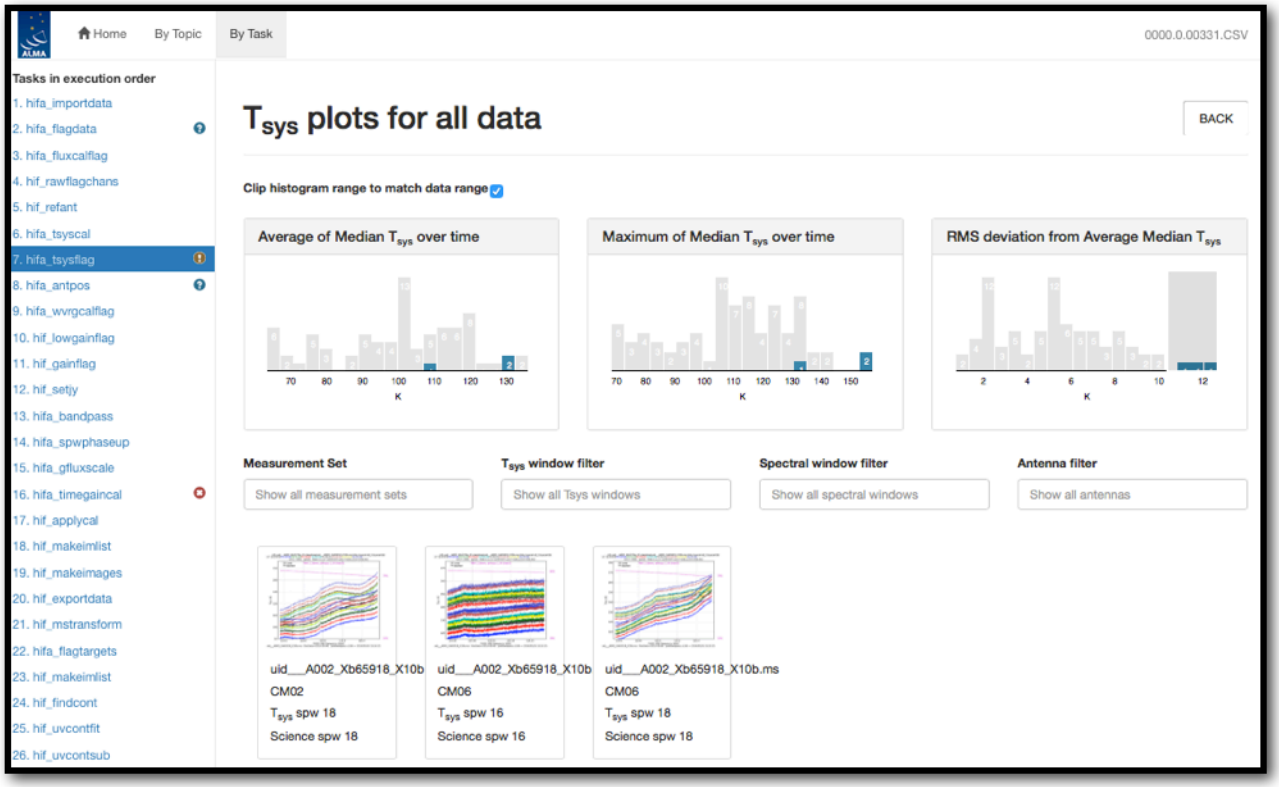


Figure 16: Same as Figure 14, but filtering to the plot of interest by using the mouse to draw a grey box on the highest histogram values in the RMS deviation from Average Median Tsys histogram plot (upper right). To clear the grey box filters on the histograms, click on any white space in the histograms.

7.7 WebLog Quality Assessment (QA) Scoring

Pipeline tasks have scores associated with them in order to quantify the quality of the dataset and the calibration. The scores are between 0.0 and 1.0 and are colorized according to the following table:

Score	Color	Comment
0.90-1.00	Green	Standard/Good
0.66-0.90	Blue	Below standard
0.33-0.66	Yellow	Warning
0.00-0.33	Red	Error

7.7.1 Interferometric Pipeline QA Scores

Pipeline Task	Pipeline QA Scoring Metric	Score
hifa_importdata	Checking that the required calibrators are present	1.0 all present
		0.1 subtracted for missing bandpass or flux calibrator
		1.0 subtracted for missing phase calibrator or Tsys calibration
		0.5 subtracted for existing processing history
hifa_flagdata	Determining percentage of incremental flagging	0 < score < 1 === 60% < fraction flagged < 5%" (for 'online', 'shadow', 'qa0', 'before' and 'applycal') where "0 < score < 1 === HIGH% < fraction flagged < LOW%" means • Score is 0 if flag fraction is >= HIGH% • Score is 1 if flag fraction is <= LOW% Score is linearly interpolated between 0 and 1 for fractions between HIGH% and LOW%
hifa_fluxcalflag	Determining percentage of incremental flagging	Additional 0%-5% flagging: score=1.0; flagging 5%-50% => 1.0...0.5; >50%: score=0.0
hif_rawflagchans	Determining percentage of data flagged due to deviant channels in rawdata	0% flagged: score=1.0; 100% flagged: score=0.0
hif_refant	Determining if a reference antenna centrally located and not flagged a lot	Sum of two scores. Score1: 1- [(distance from array center) / (distance of furthest antenna from array center)] Score2: 1- [(#good visibilities)/max(# good visibilities)]
hifa_tsysflag	Determining percentage of incremental flagging	Additional 0%-5% flagging: score=1.0; flagging 5%-50% => 1.0...0.5; >50%: score=0.0
hifa_antpos	Determining if antenna positional corrections were applied	1.0 if no corrections needed; 0.9 if one or more antennas were corrected.
hifa_wvrgcalflag	Checking phase RMS improvement	0.0 if RMS(before)/RMS(after) < 1, 0.5 ... 1.0 for ratios between 1 and 2, and 1.0 for ratios > 2
hif_lowgainflag	Determining percentage	Additional 0%-5% flagging: score=1.0; flagging

	of incremental flagging	5%-50% => 1.0...0.5; >50%: score=0.0
hif_gainflag	Determining percentage of incremental flagging	Score 1: Additional 0%-5% flagging: score=1.0; flagging 5%-50% => 1.0...0.5; >50%: score=0.0 Score 2: if a flagging view could be created then 1.0, otherwise 0.0
hifa_bandpass	Judging phase and amplitude solution flatness per antenna, spectral window and polarization (interim measure, future scoring will be based on data with solutions applied)	two algorithms: Wiener entropy and derivative deviation, and signal-to-noise ratio (scores: Wiener entropy: error function with 1-sigma deviation of 0.001 from 1.0; derivative deviation: error function with 1-sigma deviation of 0.03 for the outlier fraction; signal-to-noise ratio: error function with 1-sigma deviation of 1.0 for the signal-to-noise ratio)
hifa_spwphaseup	Determining fraction of spectral windows without phase solutions transferred from other windows	Score is the fraction of spectral windows for which phase solutions are unmapped to expected number of spectral windows
hifa_gfluxscale	Determining SNR of fitted flux values	Fitted flux values with SNR < 5.0 are assigned a score of 0.0, SNR > 20.0 a score of 1.0, and a linearly scaled value in between
hifa_timegaincal	Determining X-Y / X2-X1 phase solution deviations	Standard deviation of X-Y phase difference converted to path length: 1.0 if lower than 4.25e-6 m, 0.0 if higher than 7.955e-2 m, with an exponential decrease in between. Standard deviation of X2-X1 phase differences of subsequent integrations converted to path length: 1.0 if lower than 3.08-e-5 m, 0.0 if higher than 2.24e-2 m, with an exponential decrease in between. NB: The high limits are currently dummies. Determining their realistic values is still under development.
hifa_applycal	Determining percentage of incremental flagging	Additional 0%-5% flagging: score=1.0; flagging 5%-50% => 1.0...0.5, >50%: score=0.0
hif_makeimlist	Determine if expected targets/spw will be imaged	1.0 when all objects with desired intent appear in list for all science spw
hif_makeimages (non-checksource calibrators & science targets)	Determine if noise is close to theoretical	Ratio of sensitivity measured in non-pbcor image in a 0.3 – 0.2 PB annulus compared to 0.25 times clean threshold; Score=1 when ratio is 1 or lower Score=0 when ratio is 5 or higher
hif_makeimages (Checksources)	Determine if phase transfer worked for checksource, by checking for decorrelation and positional shift	Geometric mean of following two scores: Score1=1.0 – abs[(catalog position – fitted position)/beam size] Score2=1.0-abs[gfluxscale flux – fitted image flux)/gfluxscale flux]
hif_exportdata	Determining Pipeline products have been exported	1.0 when files successfully exported
hif_mstransform	Determine if proper files were created	1.0 when target.ms files successfully created; otherwise 0.0
hifa_flagtargets	Determine if any target flags were applied	1.0 when no flagging commands applied

hif_findcont	Determine if continuum could be identified for all spw	1.0 if continuum frequency ranges found for all spw
hif_uvcontfit	Determine if continuum could be fit	1.0 if continuum fit table created
hif_uvcontsub	Determine if continuum could be subtracted	Always set = 1.0

7.7.2 Single-Dish Pipeline QA scores

Pipeline Task	Pipeline QA Scoring Metric	Score
hsd_importdata	Checking that the required calibrators are present	1.0 ATMOSPHERE intents are present
		0.5 subtracted for existing processing history
		0.5 subtracted for existing model data
		1.0 one continuous observing session
		1.0 all source coordinate are present
hsd_flagdata	Determining percentage of incremental flagging	0 < score < 1 === 60% < fraction flagged < 5%" (for 'online', 'shadow', 'qa0', 'before' and 'applycal') where "0 < score < 1 === HIGH% < fraction flagged < LOW%" means • Score is 0 if flag fraction >= HIGH% • Score is 1 if flag fraction <= LOW% Score is linearly interpolated between 0 and 1 for fractions between HIGH% and LOW%
hsd_k2jycal	Checking that all Kelvin-to-Jy conversion factors are provided	0.0 Missing Kelvin-to-Jy conversion factor for some data
hsd_applycal	Determining percentage of incremental flagging	Additional 0%-5% flagging: score=1.0; flagging 5%-50% => 1.0...0.5, >50%: score=0.0
hsd_baseline	Checking that one or more than one emission line is detected by line-finder.	1.0 there is more than one emission line detected in at least one spw.
		0.0 No line is detected in all spw.
hsd_blflag	Determining percentage of incremental flagging per source per spw.	Additional 0%-5% flagging: score=1.0; flagging 5%-50% => 1.0...0.5, >50%: score=0.0
hsd_exportdata	Checking that the required files are exported	1.0 pipeline processing request file is exported
		1.0 pipeline WebLog file is exported
		1.0 pipeline script file is exported
		1.0 pipeline restore script file is exported
		1.0 pipeline commands log file is exported
		1.0 No missing final flag version files
		1.0 No missing final apply commands files
		1.0 No missing caltables files

8 The “By task” WebLog for Interferometric Data

This section describes navigation of the Task sub-pages for each Interferometric Pipeline task starting from the “By Task” tab. For a fuller description of each task, refer to **ALMA Science Pipeline Reference Manual**.

8.1 hifa_importdata

In this task, ASDMs are imported into measurement sets, Binary Data Flags are applied, and some properties of those MSs are calculated. The WebLog page shows a summary of imported MSs, and flux densities of calibrators. Flux densities are read from the Source table of the ASDM, which is recorded by the online system at the time of observation by interpolating in frequency the recent measurements in the calibrator catalog (see Appendix C of the ALMA C4 Technical Handbook). The flux densities for each calibrator in each science spw in each MS are written to the file flux.csv in the calibration/ subdirectory of a data delivery package. The values in this file can be edited before continuing with the pipeline execution if you first use the importonly option of eppr.executeppr.

8.2 hifa_flagdata

In this task, the online (XML format) flags, which includes the QA0 flags for antenna pointing calibration failures, are applied along with the rest of the deterministic flagging reasons (unwanted intents, autocorrelations, shadowed antennas, and TDM edge channels). The WebLog page shows the percentage of flagged data per MS. The “Before Task” column contains only the effect of the Binary Data Flags (BDF) applied during `hifa_importdata`. The additional flags are applied in the order of columns shown in the table. The percentage in each column reflects the additional amount of data flagged when applying this flag reason. The QA score for this stage is based on BDF+QA0+online+template+shadow flagging.

8.3 hifa_fluxcalflag

The WebLog shows any flagging or spwmap that was required. If the flux calibrator is a solar system object, known lines in the object (e.g. CO in Titan’s atmosphere) are flagged by this task. If more than 75% of a given spw is flagged on the flux calibrator for this reason, then a spwmap is calculated to transfer the flux scale from another spw. The WebLog shows if any flagging or spwmap was required.

8.4 hif_rawflagchans

This task was designed to detect severe baseline-based anomalies prior to performing antenna-based calibration. These bad data are often due to hardware problems during the observation. Outlier channels and outlier baselines are detected in the uncalibrated visibilities of the bandpass calibrator.

The WebLog page links to the images of the values used for flagging. Any flagged data are shown on the plots along with a summary of all flagging performed in this task. The following two rules are used to evaluate the need for flagging:

1) **"bad quadrant" matrix flagging rule:**

   This starts with the "baseline" vs. "channel" flagging view. In this view, some data points may already be flagged, e.g. due to an earlier pipeline stage.

   First, outliers are identified as those data points in the flagging view whose value deviates from the median value of all non-flagged data points by a threshold factor times the median

absolute deviation (MAD) of the values of all non-flagged data points, where the threshold is 'fbq_hilo_limit' (default: 8.0).

In formula: *flagging mask = (data - median(all non-flagged data)) > (MAD(all non-flagged data) * fbq_hilo_limit)*

Next, the flagging view is considered as split up in 4 quadrants of channels (since some problems manifest in only one or more quadrants), and each antenna is evaluated separately as follows:

a) Select baselines belonging to antenna and select channels belonging to quadrant.
b) Determine number of newly found outlier datapoints within selection.
c) Determine number of originally unflagged datapoints within selection.
d) Determine fraction of "number of newly found outliers" over "number of originally unflagged datapoints".
e) If the latter fraction exceeds the fraction threshold 'fbq_antenna_frac_limit' (default: 0.2), then a flagging command is generated that will flag all channels within the evaluated quadrant for the evaluated antenna.
f) Otherwise, no action is taken (i.e. the newly found outlier datapoints are not individually flagged by this rule),

Next, the flagging view is still considered as split up in 4 quadrants of channels, and each baseline is evaluated separately, as follows:

a) Select baseline and select channels belonging to quadrant.
b) Determine number of newly found outlier datapoints within selection.
c) Determine number of originally unflagged datapoints within selection.
d) Determine fraction of "number of newly found outliers" over "number of originally unflagged datapoints".
e) If the latter fraction exceeds the fraction threshold 'fbq_baseline_frac_limit' (default: 1.0), then a flagging command is generated that will flag all channels within the evaluated quadrant for the evaluated baseline.
f) Otherwise, no action is taken (i.e. the newly found outlier datapoints are not individually flagged by this rule).

2) "outlier" matrix flagging rule:

Data points in the flagging view are identified as outliers if their value deviates from the median value of all non-flagged data points by a threshold factor times the median absolute deviation of the values of all non-flagged data points, where the threshold is 'fhl_limit' (default: 20.0).

*In formula: flagging mask = (data - median(all non-flagged data)) > (MAD(all non-flagged data) * fhl_limit)*

Flagging commands are generated for each of the identified outlier data points.

If the number of data points in the flagging view are smaller than the minimum sample 'fhl_minsample' (default: 5), then no flagging is attempted.

8.5 hif_refant

An ordered list of preferred reference antennas is calculated, with preference given to central array location and low flagging fraction through the following score:

Refant score = [1-(normalized distance from center)] + [1-(normalized fraction of good data)]

The WebLog page shows that list, and the score for each antenna can be found in the casa log for this stage.

8.6 hifa_tsyscal

System temperature (Tsys) as a function of frequency is calculated from the atmospheric calibration scan data by the online system at the time of observation. These spectra are imported to a table of the MS during `hifa_importdata`. In `hifa_tsyscal`, these spectra are copied into a CASA calibration table by the gencal task, which flags channels with zero or negative Tsys. The WebLog shows the mapping of Tsys spectral windows to science spectral windows, and plots Tsys before flagging.

8.7 hifa_tsysflag

This task flags the Tsys cal table created by the `hifa_tsyscal` pipeline task. Erroneous Tsys measurements of several different kinds are detected, including anomalously high Tsys over an entire spectral window, spikes or “birdies” in Tsys, and discrepant “shape” or Tsys as a function of frequency. Details are provided in the WebLog for each kind of flagging performed, and all of the Tsys spectra are plotted again. In these plots, all of the anomalies should be gone.

Tsysflag provides six separate flagging metrics, where each metric creates its own flagging view and has its own corresponding flagging rule(s). In the current standard pipeline, all six metrics are active, and evaluated in the order set by the parameter "metric_order" (default: 'nmedian, derivative, edgechans, fieldshape, birdies, toomany').

1) **Metric 1: "nmedian"**

   A separate view is generated for each polarisation and each spw. Each view is a matrix with axes "time" vs. "antenna". Each point in the matrix is the median value of the Tsys spectrum for that antenna/time.

   The views are evaluated against the "nmedian" matrix flagging rule, where data points are identified as outliers if their value is larger than a threshold-factor * median of all non-flagged data points, where the threshold is 'fnm_limit' (default: 2.0).

   Flagging commands are generated for each of the identified outlier data points.

2) **Metric 2: "derivative"**

   A separate view is generated for each polarisation and each spw. Each view is a matrix with axes "time" vs. "antenna". Each point in the matrix is calculated as follows:
   - calculate "valid_data" as the channel-to-channel difference in Tsys for that antenna/timestamp (for unflagged channels)
   - calculate *median(abs(valid_data - median(valid_data))) * 100.0*

   The views are evaluated against the "max abs" matrix flagging rule, where data points are identified as outliers if their absolute value exceeds the threshold "fd_max_limit" (default: 5).

   Flagging commands are generated for each of the identified outlier data points.

3) **Metric 3: "edgechans"**

   A separate view is generated for each spw and each of these intents: ATMOSPHERE, BANDPASS, and AMPLITUDE. Each view contains a "median" Tsys spectrum where for each channel the value is calculated as the median value of all selected (spw,intent) Tsys spectra in that channel (this combines data from all antennas together).

   The views are evaluated against the "edges" vector flagging rule, which flags all channels from the outmost edges (first and last channel) until the first channel for which the channel-to-channel difference first falls below a threshold times the median channel-to-channel difference, where the threshold is "fe_edge_limit" (default: 3.0).

A single flagging command is generated for all channels newly identified as "edge channels".

4) **Metric 4: "fieldshape"**

A separate view is generated for each spw and each polarization. Each view is a matrix with axes "time" vs. "antenna". Each point in the matrix is a measure of the difference of the Tsys spectrum for that time/antenna from the median of all Tsys spectra for that antenna/spw in the "reference" fields that belong to the reference intent specified by "ff_refintent" (default: "BANDPASS").

The exact fieldshape value is calculated as: 100 * mean(abs(normalized tsys - reference normalized tsys)), where a 'normalized' array is defined as: "array / median(array)"

The views are evaluated against the "max abs" matrix flagging rule, where data points are identified as outliers if their absolute value exceeds the threshold "ff_max_limit" (default: 5).

5) **Metric 5: "birdies"**

A separate view is generated for each spw and each antenna. Each view contains a "difference" Tsys spectrum calculated as:

"channel-by-channel median of Tsys spectra for antenna within spw" - "channel-by-channel median of Tsys spectra for all antennas within spw".

The views are evaluated against the "sharps" vector flagging rule, which flags each view in two passes:

a. flag all channels whose absolute difference in value to the following channel exceeds a threshold "fb_sharps_limit" (default: 0.05).

b. around each newly flagged channel, flag neighboring channels until their channel-to-channel difference falls below 2 times the median channel-to-channel difference (this is intended to flag the wings of sharp features).

A single flagging command is generated for all channels newly identified as "birdies".

6) **Metric 6: "toomany"**

A separate view is generated for each polarisation and each spw. Each view is a matrix with axes "time" vs. "antenna". Each point in the matrix is the median value of the Tsys spectrum for that antenna/time. (This is the same as for "nmedian" metric).

The views are evaluated against two separate flagging rules:

a. "tmf" (too many flags): This evaluates each timestamp one-by-one, flagging an entire timestamp when the fraction of flagged antennas within this timestamp exceeds the threshold "tmf1_limit" (default: 0.666). Flagging commands are generated per timestamp.

b. "tmef" (too many entirely flagged): This evaluates all timestamps at once, flagging all antennas for all timestamps within current view (spw, pol) when the fraction of antennas that are entirely flagged in all timestamps exceeds the threshold "tmef1_limit" (default: 0.666). Flagging commands are generated for each data point in the view that is newly flagged.

8.8 hifa_antpos

Sometimes the antenna positions were refined after the science data were recorded. If such refinements have been located, they are applied in this task. The corrections are listed in the WebLog, and the uvw values for the visibility data are recalculated.

8.9 hifa_wvrgcalflag

Water Vapor Radiometer (WVR) power measurements are converted into a phase correction table that can be applied to the science data. The phase rms during observation of the bandpass calibrator, with and without the WVR correction, is used 1) to detect poorly performing WVR units on individual antennas, and 2) to determine of the WVR correction helps overall.

The WebLog shows the effects of the phase correction in several ways, if any antennas' WVR data are flagged (the required phase correction is then interpolated from nearby antennas), and also prints a warning of the correction is deemed not helpful enough to apply at all.

8.10 hif_lowgainflag

Antennas with persistently low amplitude gains are detected and flagged. The WebLog links to grayscale images of the relative gain of each antenna calculated using the observation of the bandpass calibrator, and shows if any antennas are flagged.

This task first creates a bandpass caltable, then a gain phase caltable, and finally a gain amplitude caltable. This final gain amplitude caltable is used to identify antennas with outlier gains, for each spw. Flagging commands for outlier antennas (per spw) are applied to the entire MS.

A separate view is created for spw. Each view is a matrix with axes "time" vs. "antenna". Each point in the matrix is the absolute gain amplitude for that antenna/timestamp.

The views are evaluated against the "nmedian" matrix flagging rule, where data points are identified as outliers if:

a. Their value is smaller than a threshold-factor * median of all non-flagged data points, where the threshold is 'fnm_lo_limit' (default: 0.7), or

b. Their value is larger than a threshold-factor * median of all non-flagged data points, where the threshold is 'fnm_hi_limit' (default: 1.3).

Flagging commands are generated for each of the identified outlier data points.

8.11 hif_gainflag

Antennas whose gain as a function of time shows anomalously high scatter are detected and flagged. The WebLog links to grayscale images of the gain rms per antenna, showing any that are flagged.

This task first creates a phased-up bandpass caltable, then a gain phase caltable, and finally a gain amplitude caltable. This final gain amplitude caltable is used to identify antennas with outlier gains, for each spw. Flagging commands for outlier antennas (per spw) are applied to the entire MS.

Gainflag offers two separate flagging metrics, where each metric creates its own flagging view and has its own corresponding flagging rule(s). In the Cycle 4 pipeline release, only the "rmsdeviant" metric is active. This works as follows:

a. A separate view is created for each spw. Each view is a matrix with axes "time" vs. "antenna". Each point in the matrix is the "standard deviation of the gain amplitudes for that antenna and all timestamps, divided by the median absolute deviation of the gain amplitude for all antennas and all timestamps".

b. The views are evaluated against the "max abs" matrix flagging rule, where data points are identified as outliers if their absolute value exceeds the threshold "frmsdev_limit" (default: 3.5).

Flagging commands are generated for each of the identified outlier data points

8.12 hif_setjy

The model flux density of the amplitude calibrator is set, either from an internal CASA model (solar system objects), or the results of observatory calibrator monitoring (quasars) which ultimately appear in the file flux.csv (see `hifa_importdata`). These flux densities are listed on the WebLog page, along with plots of the amplitude calibrator as a function of uv distance (which is useful to assess resolved solar system objects).

8.13 hifa_bandpass

In this task, the bandpass calibrator is self-calibrated (phase only is first calibrated on as short a time interval as allowed by signal-to-noise, listed on the WebLog page). The antenna-based bandpass phase and amplitude solution is then calculated using a S/N-dependent frequency interval, also listed on the WebLog page. Finally, the WebLog page links to plots of all of the bandpass solutions, with the atmospheric transmission curve overlaid.

8.14 hifa_spwphaseup

The relative phase offsets between spectral windows are determined for each antenna using the observation of the bandpass calibrator. (The offset is assumed to be constant in time during each execution.) If narrow spectral windows are present, a mapping is determined so that the calculated phase calibration as a function of time can be subsequently transferred (during subsequent gaincal and applycal tasks) from wider, higher S/N spectral windows to the narrow ones. If any such reference spwmaps are required, then they are listed on the WebLog page.

8.15 hifa_gfluxscale

In this task, the absolute flux scale is transferred from the amplitude calibrator to the other calibrators and ultimately to the science target (via the phase calibrator). A phase-only self-calibration is performed on all calibrators prior to this flux calculation.

The WebLog for this stage lists the derived flux densities of the non-amplitude calibrators (usually phase and bandpass calibrators), along with the flux values extracted from the ALMA Source Catalog. Plots of amplitude as a function of uv distance are shown, and If the absolute flux calibrator is resolved (decreasing flux with increasing uv distance, usually only the case for solar system objects), only data on short baselines are used to calculate the flux densities of the secondary calibrators. Any such uv limits are listed in the table at the top of the WebLog page.

8.16 hifa_timegaincal

In this task, gain as a function of time is calculated from observations of the phase calibrator. The WebLog page shows plots of this gain, both on a scan timescale (as will be interpolated to the science target), and on an integration timescale (useful for assessing weather and calibration quality).

8.17 hif_applycal

In this task, all previously calculated calibration tables are applied to the science data. Any failed calibration solutions, and flagged Tsys scans, will result in flagging of actual science data in this stage, so the WebLog shows a summary of that additional flagging, and high flagging will result in a low QA score. The WebLog page also includes many useful plots of the calibrated data as a function of time and frequency. Outliers in these plots can indicate any remaining bad data.

8.18 hif_makeimlist: Set-up parameters for calibrator images

This stage determines image parameters (image size, cell size, etc) to be used in the subsequent `hif_makeimages` stage, and reports them on the WebLog page (See Figure 17). The "specmode" can be mfs for per-spw continuum multi-frequency synthesis images, "cont" for mfs continuum images of several spectral windows, or "cube" for spectral cubes. The first time the task is run is in preparation for making per-spw mfs images of the calibrators.

18. Make image list

Set-up image parameters for calibrator imaging

BACK

List of Clean Targets

field	intent	spw	phasecenter	cell	imsize	imagename	specmode	start	width	nbin	nchan	uvrange
J1256-0547	BANDPASS	16	ICRS 12:56:11.1670 -005.47.21.525	['2.5arcsec']	[60, 60]	uid___A002_Xb5270a_X1c.sSTAGENUMBER.J1256-0547_bp.spw16.mfs	mfs			-1	-1	
J1256-0547	BANDPASS	18	ICRS 12:56:11.1670 -005.47.21.525	['2.5arcsec']	[60, 60]	uid___A002_Xb5270a_X1c.sSTAGENUMBER.J1256-0547_bp.spw18.mfs	mfs			-1	-1	
J1256-0547	BANDPASS	20	ICRS 12:56:11.1670 -005.47.21.525	['2.5arcsec']	[60, 60]	uid___A002_Xb5270a_X1c.sSTAGENUMBER.J1256-0547_bp.spw20.mfs	mfs			-1	-1	
J1256-0547	BANDPASS	22	ICRS 12:56:11.1670 -005.47.21.525	['2.5arcsec']	[60, 60]	uid___A002_Xb5270a_X1c.sSTAGENUMBER.J1256-0547_bp.spw22.mfs	mfs			-1	-1	
J1316-3338	PHASE	16	ICRS 13:16:07.9859 -033.38.59.173	['2.5arcsec']	[60, 60]	uid___A002_Xb5270a_X1c.sSTAGENUMBER.J1316-3338_ph.spw16.mfs	mfs			-1	-1	
J1316-3338	PHASE	18	ICRS 13:16:07.9859 -033.38.59.173	['2.5arcsec']	[60, 60]	uid___A002_Xb5270a_X1c.sSTAGENUMBER.J1316-3338_ph.spw18.mfs	mfs			-1	-1	
J1316-3338	PHASE	20	ICRS 13:16:07.9859 -033.38.59.173	['2.5arcsec']	[60, 60]	uid___A002_Xb5270a_X1c.sSTAGENUMBER.J1316-3338_ph.spw20.mfs	mfs			-1	-1	
J1316-3338	PHASE	22	ICRS 13:16:07.9859 -033.38.59.173	['2.5arcsec']	[60, 60]	uid___A002_Xb5270a_X1c.sSTAGENUMBER.J1316-3338_ph.spw22.mfs	mfs			-1	-1	

Clean Targets Summary

Figure 17: Example of the WebLog for the `hif_makeimlist` stage. This example is for setting up the parameters for calibrator per-spw mult-frequency synthesis (mfs) continuum images.

8.19 hif_makeimages: Make calibrator images

This stage actually creates the images requested by the most recent `hif_makeimlist`. The first time it is run is to create per-spw mfs continuum images of the calibrators. See the description in Sec. 8.28 for more information and examples of the `hif_makeimages` stage. Low QA scores for non-Check source calibrators may indicate the need for additional flagging.

8.20 hif_checkproductsize: Mitigation to avoid overly long runs

One of the more significant new behaviors of the Cycle 4 "patch" version of the Pipeline is the inclusion of an automated image size mitigation function, hif_checkproductsize, that is run before the calibration hif_exportdata stage. This function will modify the characteristics of the imaging products in order to decrease their size, thereby decreasing the time needed to make them so that data can be delivered to PI's more expediantly. Figure 18 shows an example weblog page for a mitigated dataset. Datasets that have been mitigated will have imaging products with different characteristics than those that have not been mitigated. Full imaging products can be recreated by users, using the tclean commands that are in the casa_commands.log file, or by calling the appropriate hif_makeimlist, hif_makeimages with the defaults (which will make full imaging products without mitigations – be aware that this could take many days to complete).

The mitigations are done in a priority order, with the mitigation halted once the predicted sizes fall below the thresholds. For the Cycle 4 patch, the thresholds are: any individual cube with MaxCubeSize greather than 30GB, or a total product with MaxProductSize greater than 400 GB. The mitigation then proceeds according to the following logic:

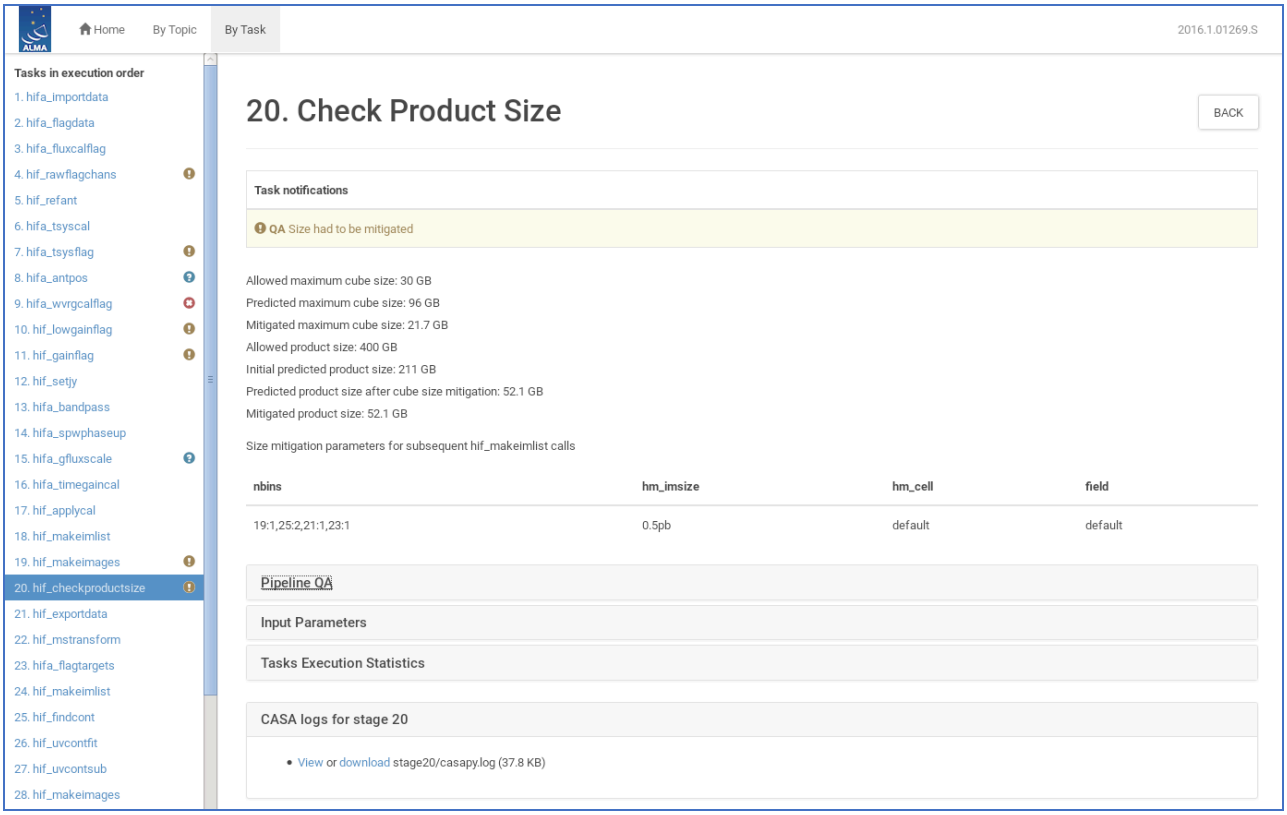


Figure 18: Screenshot of new hif_checkproductsize stage of IF Pipeline. In this example, the cubes for spw25 had to be binned by a factor of 2, and the FOV was limited to the 0.5 response point of the primary beam in order to get the products below the default thresholds. Before the mitigation the maximum cube would have been 96GB; after the mitigation, it is predicted to be 21.7 GB.

1. Cube Size Check: Any cube estimated to be > MaxCubeSize?
    1. If no, go to Product Size Check
    2. If yes, recalculate max(estimated cube size) after each of the following modifications, and go to Product Size Check when max(estimated cube size)<MaxCubeSize
        1. Any spw≥3840? If so, binFactor=2 for those spw. If estimate still > MaxCubeSize, then:
        2. Any online unbinned spw=1920? If so, binFactor=2 for those spw. If estimate still > MaxCubeSize, then:
        3. For single pointing MOUS, if (estimate/ MaxCubeSize) < 1.4 then image out to 0.3PB instead of 0.2PB (and change noise annulus correspondingly), else image out to 0.5 PB (and change noise annulus correspondingly). If estimate still > MaxCubeSize, then:
        4. Use a cellsize of 3 pixel/beam instead of 5.
        5. If estimate still > MaxCubeSize, produce “x” error for this stage with comment *“Error! Product size cannot be mitigated”* & report remaining mitigation factor needed. Exit at next hif_makimlist stage with error *“Size mitigation had failed. Will not create any clean targets”.*
2. Product Size Check: recalculate total size given all previous steps. Is > MaxProductSize?
    1. If no, then go on to imaging.
    2. If yes, is [estimated product size after Step 1 / MaxProductSize] / N_sources > 1, if yes, then:
    3. Produce “x” error for this stage with comment *“Error! Product size cannot be mitigated”* & report remaining mitigation factor needed. Exit at next hif_makimlist stage with error *“Size mitigation had failed. Will not create any clean targets”.*

4. Else, set science target image list to only image the first N=INT(MaxProductSize/ [estimated product size after Step 1 / N_sources]).

In the example shown in Figure 18, the initial data products were estimated to include a cube that would be 96 GB. This triggered two mitigations: spectral window 25 was binned by a factor of 2, and the FOV was restricted to the 0.5 response point of the Primary Beam. This was sufficient to get the cube size down to 21.7 GB, so the mitigation cascade stopped. The total product size after the cube mitigation is 52.1 GB, so products for all sources could be made.

8.21 hif_exportdata

Calibration tables, calibrator images (exported in fits format), and other products are moved from the pipeline /working to the /products directory.

NOTE: The subsequent stages are only present if the imaging pipeline was run.

8.22 hif_mstransform

For each execution, calibrated visibilities for the science target(s) are split to a new MS with "target.ms" in the name, as listed on the front WebLog page.

8.23 hifa_flagtargets

Flagging of the science target data, if determined to be necessary by an observatory scientist, is performed as listed in the ***flagtargetstemplate.txt** files linked to the WebLog page. The WebLog also shows a summary table of any flagging performed.

8.24 hif_makeimlist: Set-up parameters for target per-spw continuum imaging

Imaging parameters are determined and listed for creation of per-spw mfs continuum images of each science target. This run of `hif_makeimlist` also controls the parameters used to create the dirty cubes used by the `hif_findcont` stage, including any channel binning (listed in the "nbins" column of the `hif_makeimlist` table).

8.25 hif_findcont

In this task, dirty image cubes are created for each spectral window of each science target. The cubes are made at the native channel resolution unless the nbins parameter was used in the preceding `hif_makeimlist` stage. The signal as a function of frequency is determined and plotted on the WebLog page as a spectrum (see examples in Figure 19). This is not simply a mean spectrum of the target, but rather a quantity that is more sensitive to spectral features. In most cases, it is the mean spectrum above a S/N level that depends on the number of channels. If no features are found, the spectrum is recomputed over the central area of the cube in search of faint compact emission. An alternative algorithm (the per-channel peak / MAD) is invoked in certain situations, including when significant atmospheric lines are present. In either case, frequency ranges are calculated that are the least likely to contain any line emission or absorption, and these are listed in the LSRK frame on the WebLog page, as well as being indicated by the cyan colored horizontal line(s) on the spectra.

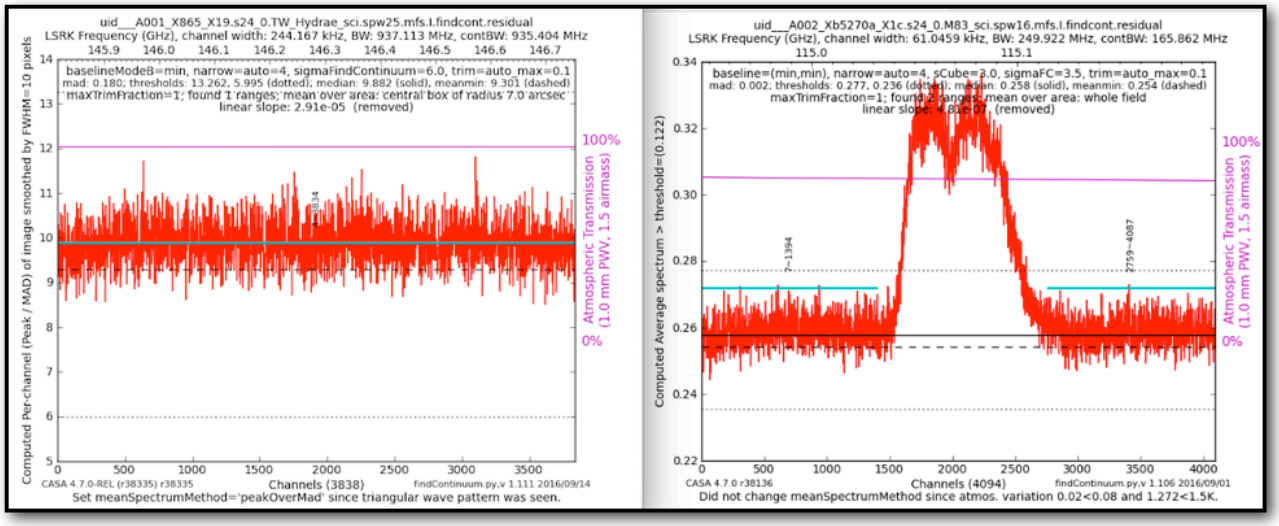


Figure 19: Two examples of `hif_findcont` plots, one with the entire window identified as continuum (left), and another with two identified continuum regions (right; identified continuum indicated by cyan lines).

The continuum frequency ranges are also printed to a file called "cont.dat". If this file already exists before `hif_findcont` is executed, then it will first examine the contents. For any spw that already has frequency ranges defined in this file, it will not perform the analysis described above in favor of the a priori ranges. For spws not listed in a pre-existing file, it will analyze them as normal and update the file. In either case, the file cont.dat is used by the subsequent `hif_uvcontfit` and `hif_makeimages` stages.

8.26 hif_uvcontfit

The previously determined continuum frequency ranges as shown in the **cont.dat** file are used to fit the continuum of each visibility. The fit is performed for each spw independently using a fitorder=1, and a calibration table is used to store the resulting fits called "uvcont.tbl". The WebLog for this stage reports the continuum ranges from `hif_findcont` in LSRK but translated to the topocentric (TOPO) frame for each MS.

8.27 hif_uvsub

The `hif_uvcontfit` calibration table is applied to the data. After this step, the original continuum + line emission is contained in the DATA column of the MS, while the continuum subtracted data are written to the CORRECTED column.

8.28 hif_makeimages: Make target per-spw continuum images

Cleaned continuum images are created for each spectral window, each science target, using the continuum frequency ranges determined from `hif_findcont` (as written in the cont.dat file). For Cycle 4, the pipeline uses a generic clean mask corresponding to everything within the 0.3 response level of the primary beam, and a conservative clean threshold of 4 x (predicted rms noise) x (dynamic range correction factor). The dynamic range correction factor accounts for the fact that sources with a high dynamic range will have larger imaging artifacts, which should not be cleaned. The artifacts are worse for poorer UV coverage, so different DR corrections factors are adopted for 12-m Array and 7-m Array observations, according to the following table:

Source Dynamic Range	12-m Array DR correction factor	Source Dynamic Range	7-m Array DR correction factor
≤ 20	1	≤ 4	1
20 – 50	1.5	4 – 40	1.5
50 – 100	2	10 – 20	2
100 – 150	2.5	20 – 30	2.5
≥ 150	max (2.5, DR/150)	≥ 30	max (2.5, DR/30)

The resulting non-primary beam corrected images are displayed on the WebLog page. For each image, the properties are shown next to the associated image png (see Figure 20). In particular, the following are reported: the center frequency, beam parameters (major and minor FWHM resolution & position angle), theoretical sensitivity, cleaning threshold, dynamic range of the dirty image (image peak to theoretical noise) and corresponding DR correction factor, the non-pbcor image rms (the noise measured in the non-primary beam corrected image over an annulus between the 0.3 to 0.2 response point of the primary beam), image max /min of the primary beam corrected image, fractional bandwidth, aggregate bandwidth, and the image QA score (meant to indicate how close the measured noise is to the theoretical noise, considering also the DR correction factor – see Sec. 7.7.1).

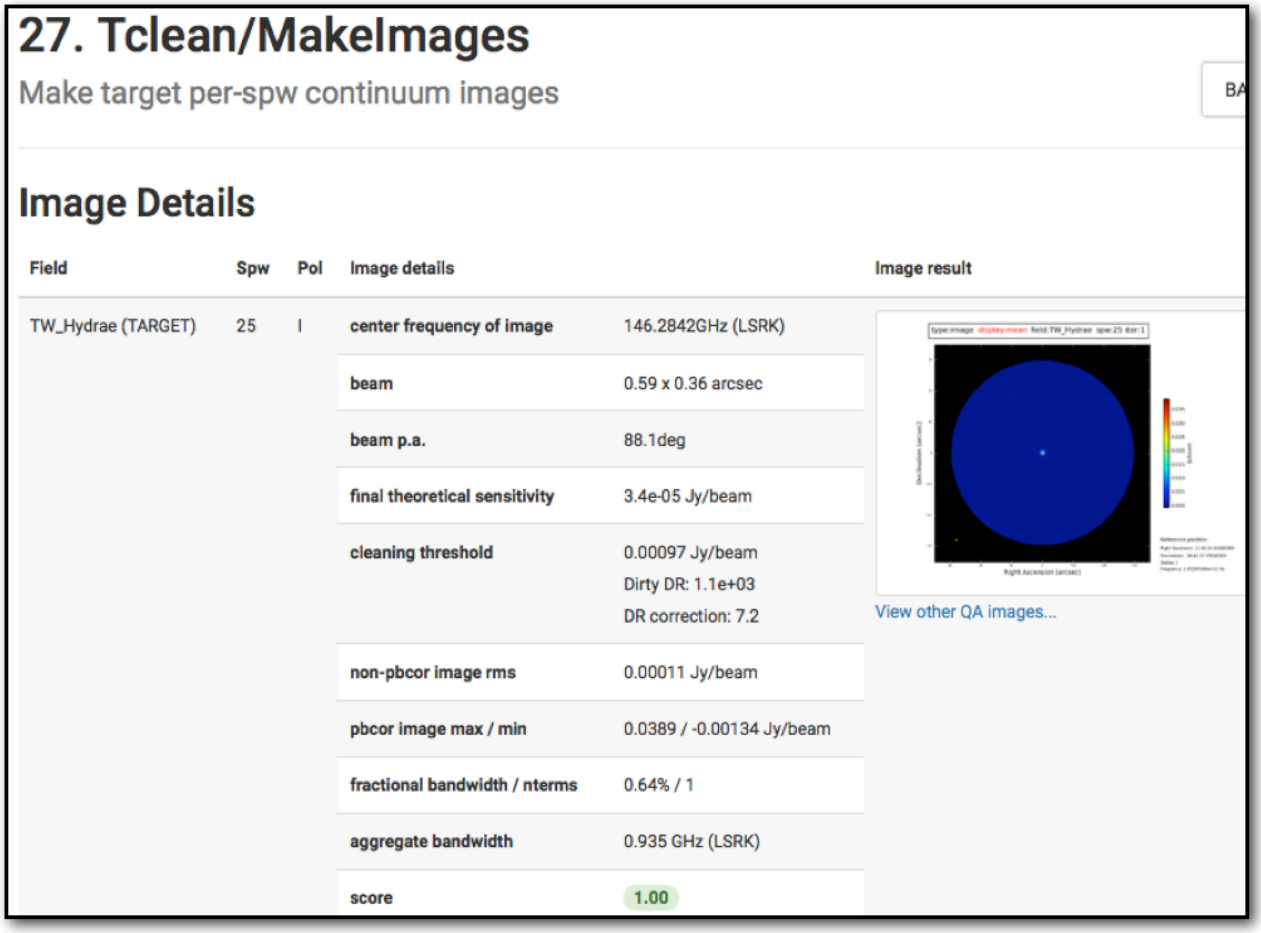


Figure 20: Example of `hif_makimages` WebLog page for per-spw images. Clicking on the thumbnail will enlarge the image. Clicking on the "View other QA images" link will bring up the detailed image page (Figure 21).

The "View Other QA Images' links for each image show the primary beam corrected image, residual, clean mask (red area), dirty image, primary beam, psf, and clean model (Figure 21).

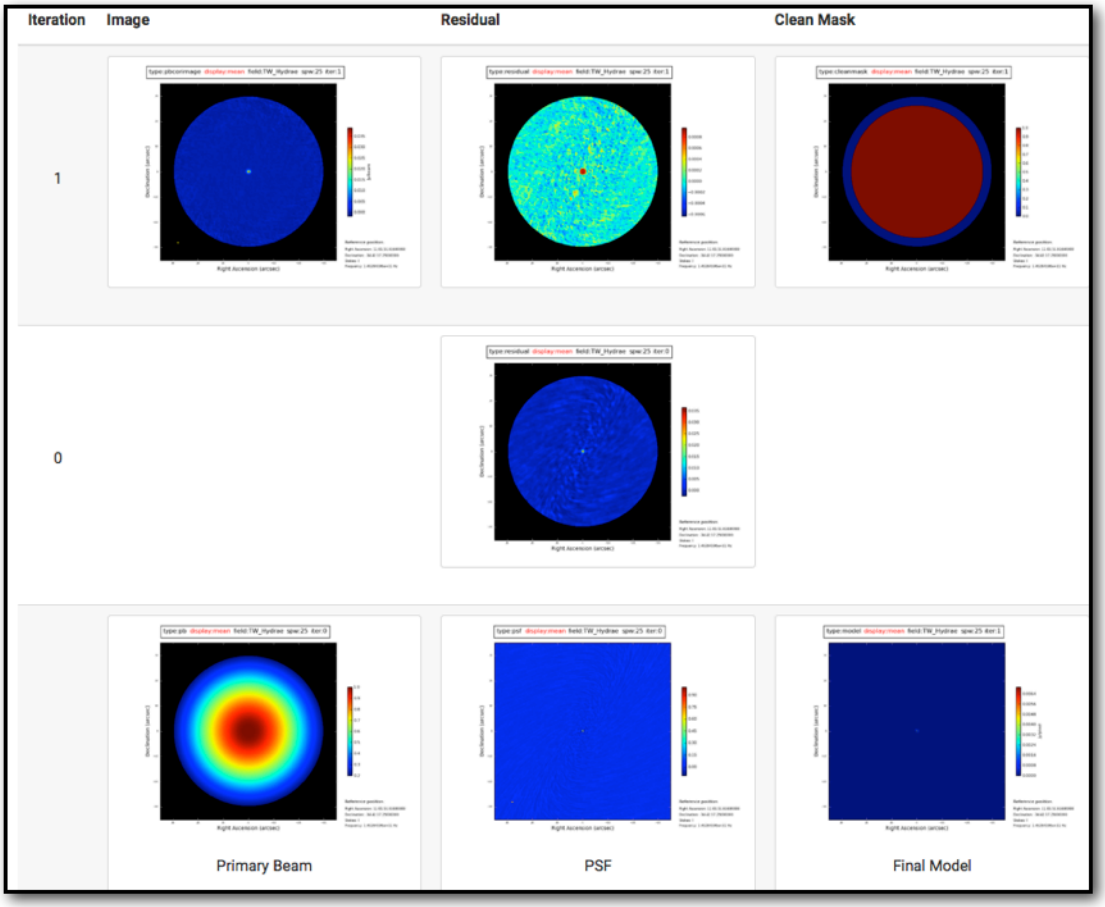


Figure 21: Details page that is displayed after clicking on the "View other QA images" link on the `hif_makimages` WebLog page.

8.29 hif_makeimlist: Set-up parameters for target aggregate continuum images

Imaging parameters are calculated and listed for creation of an aggregate (all spectral windows combined) continuum image (specmode='cont') of each science target.

8.30 hif_makeimages: Make target aggregate continuum images

A cleaned aggregate continuum image of each science target is formed from the `hif_findcont` channels (as listed in the cont.dat file) is created. The aggregate continuum image(s) are made with nterms=2 if the fractional bandwidth is ≥ 10% (only currently possible for ALMA Bands 3 and 4 data). The resulting non-primary beam corrected images are displayed on the WebLog page. The "View Other QA Images" links show the primary beam corrected image, psf, clean model, dirty image, and residual image (Figure 21).

8.31 hif_makeimlist: Set-up image parameters for target cube imaging

Parameters are calculated and listed for creation of spectral cube images of each continuum-subtracted spectral window of each science target.

8.32 hif_makeimages: Make target cubes

Cleaned continuum-subtracted cubes are created for each science target and spectral window at the native channel resolution (unless channel binning has been selected using nbins in the preceding `hif_makeimlist`) from the CORRECTED column. Cubes are made in the radio LSRK frequency frame. Only channels that have not been designated as continuum channels

are cleaned. The WebLog page displays non-primary beam corrected peak intensity images for each cube (“moment 8”) along with properties of the cubes (see Figure 22). The information is similar to that described in Sec. 8.28 for continuum images, except that the noise is the median rms over all channels (still measured in a 0.3 – 0.2 PB annulus), and instead of fractional and aggregate bandwidth the “channel” information is given as the number of channels imaged times the channel width. Recall that if no online or nbins (pipeline option) channel averaging is done, the velocity resolution will be twice the channel width.

Figure 22: Example of `hif_makimages` WebLog page for image cubes.

In addition to the “View other QA images” for continuum images demonstrated for Stage 27, an additional plot is included for continuum subtracted cubes: an integrated intensity (“moment 0”) image using the `hif_findcont` continuum frequency ranges (labeled “Line-free Moment 0”; see Figure 23), which should be noise-like if the continuum subtraction worked well.

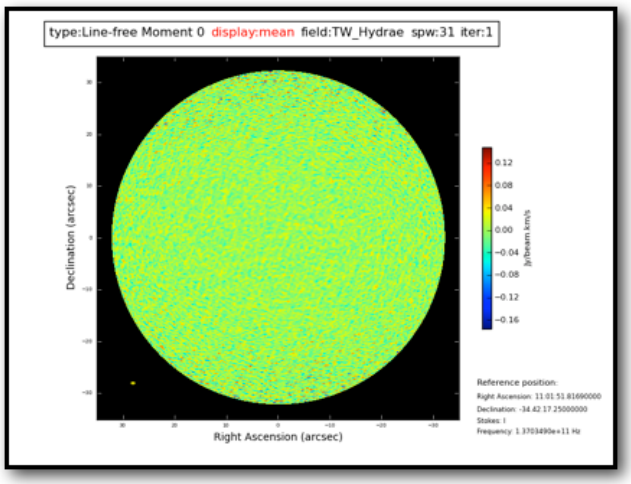


Figure 23: Example of a line-free moment 0 map shown on the details page of image cubes.

8.33 hif_exportdata

Science target images are converted to fits format and copied to the /products subdirectory as well as the cont.dat file from the `hif_findcont` stage. This stage is run in operations, but is not included in the **casa_pipescript.py** script.

9 The “By task” WebLog for Single-Dish Data

This section describes navigation of the Task sub-pages for each Single Dish Pipeline task starting from the “By Task” tab. For a fuller description of each task, refer to the **ALMA Science Pipeline Reference Manual**.

9.1 hsd_importdata

The WebLog for `hsd_importdata` task shows the summary of imported MSs, grouping of spws to be reduced as a group[1], and spw matching between Tsys and science spws. This task also generates figures of Telescope Pointings, which are available in the MS Summary page (i.e. from the Home page, click the MS name, and then click on “Telescope Pointing”). There are two types of plots that can be found containing full information on all pointings and just on-source pointings (Figure 24). In these plots, the red circle indicates the beam size of the antennas and its location is the starting position of the raster scan. The Red (small) dot indicates the last position of the raster. The green line represents the antenna slewing motion, and in the right panel of Figure 24 the green line going to/from the red dot indicates that the antenna goes to the last scan and returns to the OFF position. The grey dots indicate flagged data.

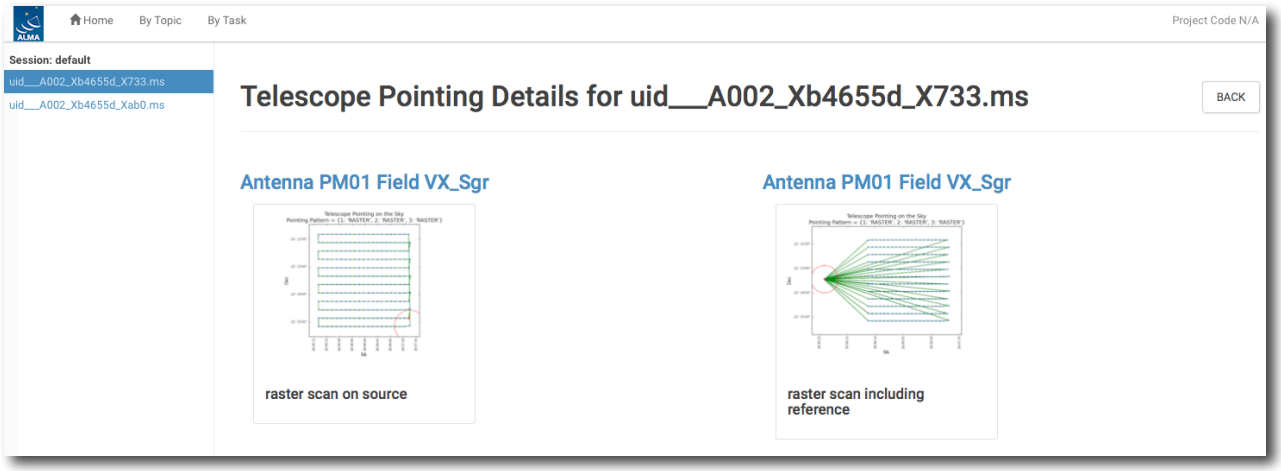


Figure 24: The detailed page of Telescope Pointing in the MS summary page.

9.2 hsd_flagdata

The WebLog for the `hsd_flagdata` task shows the summary of flagged data percentage per MS due to binary data and online flagging, manually inserted file (***flagtemplate.txt**), shadowing, unwanted intents, and edge channels. Note that the value in the “Before Task” column corresponds to the percentage of flagged data by binary data flagging.

9.3 hifa_tsyscal

This page shows the associations of Tsys and science spectral windows to be used for Tsys (amplitude-scale) calibration, and also shows the original Tsys spectra per spectral window.

9.4 hifa_tsysflag

This page shows the flagged Tsys spectra per spectral window after heuristic flagging is applied.

[1] When several ASDMs are processed at once, the Pipeline needs to group their respective spws, based on spw Name.

9.5 hsd_skycal

The WebLog shows the integrated OFF spectra per spw and per source. The y-axis is the direct output from the correlator, which means the values are dominated by signals from both the atmosphere and receivers (Figure 25). The different colors indicate different scans (times).

The time-averaged plots of the OFF spectra are also shown in this page for the purpose of assessing the time variability of the spectra. The different colors here indicate different spws. Note that the OFF spectrum is not averaged over the spectral windows yet, but it will be in the future.

The coordinates of the OFF position can be confirmed in the Reference Coordinates table.

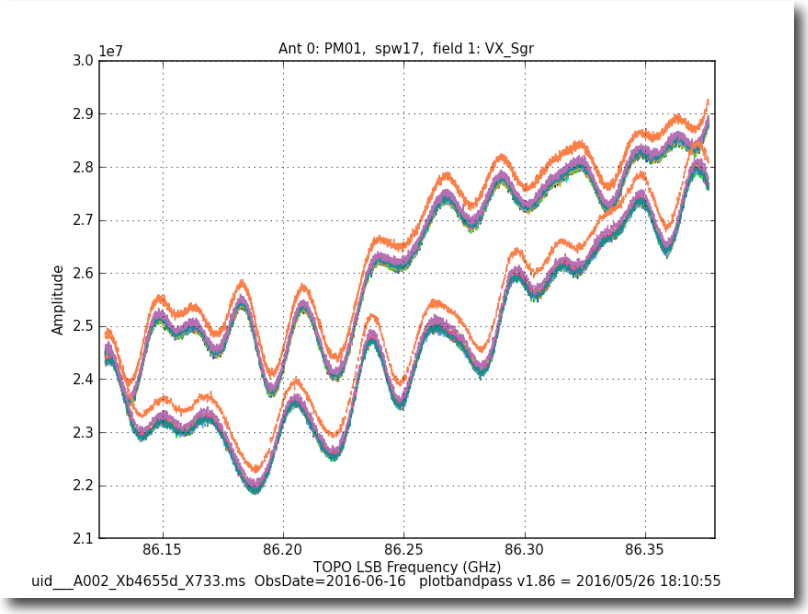


Figure 25: An example of OFF spectrum

9.6 hsd_k2jycal

This page shows the list of Kelvin to Jy conversion factors that Pipeline has read from a file "jyperk.csv", which shall contain the factors per spw, per antenna, and per polarization.

9.7 hsd_applycal

This page shows a list of the calibrated MSs with the name of the applied Tsys, Sky and amplitude calibration (Kelvin to Jy conversion) tables, and also shows the integrated spectra after calibration.

9.8 hsd_baseline

Spectral data before/after baseline subtraction

The `hsd_baseline` page of the WebLog show the grid of spectra before (top) and after (bottom) baseline subtraction (Figure 26). The plots on the `hsd_baseline` summary page (just after clicking the `hsd_baseline` link of the WebLog) show a representative spectral map for each spw out of all maps in the detail pages (when you click the "Spectral Window" link below the grid of spectra in the summary page). The detail page has similar plots but this time for each ASDM, antenna, and polarization.

On the top panel of each grid of spectra, a spatially integrated spectrum per ASDM, antenna, spw and polarization is shown. The magenta lines indicate the atmospheric transmission at each

frequency. The cyan filled regions indicate the mask channels containing emission line that are identified in the entire map, and red thick bars indicate the channels masked by a “deviation mask” algorithm, designed to exclude atmospheric lines and lines at the band edge from the baseline fit.

Below the top panel, there is a grid of spectra aligned along R.A./Decl. coordinates. Each small panel shows **one** representative spectrum per grid cell (which sometimes we call “sparse profile map”). The red (horizontal) line over-plotted on the spectrum indicates the fitted function to be used for baseline subtraction for spectral data before baseline subtraction, while the zero-level for spectral data after baseline subtraction.

Figure 26 An example of the summary page of hsd_baseline.

R.A. vs Dec. plots

There are four different plots per spw, i.e. “clustering_detection”, “clustering_validation”, “clustering_smoothing”, and “clustering_final”. The number of plots in each figure is the same as that of the candidate line components. The “cluster_detection” plot (Figure 27a) shows the grid cells having emission line exceeding the threshold. In the plot, yellow grid cells show a region where there is a single time-domain group with detected emission lines. Cyan squares indicate grid cells where there are more than one time-domain groups with detected emission lines.

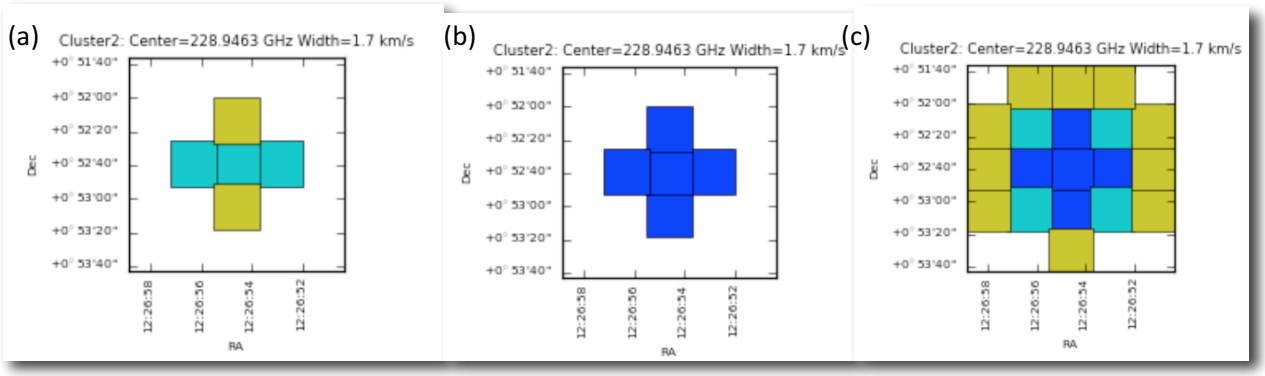


Figure 27 Examples of (a) clustering_detection, (b) clustering_validation, and (c)clustering_smoothing.

After line detection, the algorithm calculates how many spectra containing emission lines are included in the grid cell in order to judge whether the grid cell possibly contains true emission lines. At this line detection validation step, the ratio of the number of spectra having detected emission lines (defined as “Nmember”) per grid cell and the number of total spectra belonging to the grid cell (“Nspectra”) is calculated. The “clustering_validation” plot (Figure 27b) shows this ratio for each grid cell, i.e., the grid cell is marked as:

- “Validated” if Nmember/Nspectra > 0.5 (Blue squares in Figure 27b)
- “Marginally validated” if Nmember/Nspectra > 0.3 (Cyan squares)
- “Questionable” if Nmember/Nspectra > 0.2 (Yellow squares)

After the validation step, the grid containing the Nmember/Nspectra rate per grid cell is smoothed by a Gaussian-like grid function. This is to eliminate the isolated grid cells having a single emission line candidate while enhancing the grid cells with detected emission line in neighboring grid cells.

Figure 27c shows an example of “clustering_smoothing”. Blue squares represent the grid cells with points exceeding the defined threshold, i.e., the grid cells having promising detections of emission lines that are also found in the neighboring grid cells. Cyan and yellow squares are the grid cells with points slightly below the threshold (Border), or lower than the threshold (Questionable).

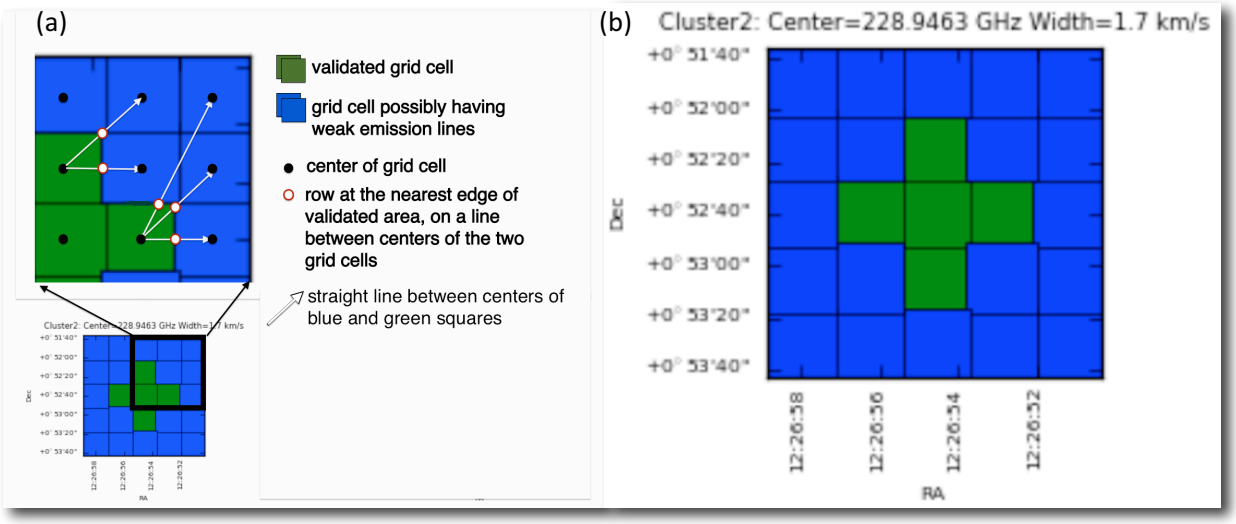


Figure 28 (a) An example of how the mask range is calculated. In the blue squares, the mask channel range is the range obtained at the nearest edge of any validated area by interpolation mask channel ranges in the validate grid cells (white-filled red circle). (b) An example of clustering_final.

As a final step, the mask region for each grid cell is determined. In the validated area after the validation and the smoothing steps (blue squares in Figure 27c or green squares in Figure 28), mask channel ranges are calculated over the spatial domain by inter/extrapolating the mask ranges of the integrated spectra in the validated cells, and over each single non-integrated spectrum. The mask channel range is determined and used in baseline subtraction in the green and blue squares of Figure 28a. An example of "clustering_final" is shown in Figure 28b.

Line Center vs. Line Width plot

This plot shows the extent of each identified emission line candidate on the parameter space of the line width versus the line center. The small dots indicate spectra containing identified emission line. The red ovals show each clustering region with a size of the cluster radius.

Number of Clusters vs. Score plot

This plot shows the number of clusters and corresponding scores based on the cluster size determined from the "line width" v.s. "line center" plot using clustering analysis (K-means algorithm). The scoring is empirically defined so that the score gets better (smaller) when the cluster size is smaller, the number of clusters is smaller, and the number of outliers is fewer than those of other clusters. The users will know which number of clusters is more plausible by searching for the number of clusters with a lower score. This plot is basically for developers.

9.9 hsd_blflag

The WebLog shows the list of flagged data percentage using five criteria that are explained in the ALMA Pipeline Reference Manual. When you click on "details", you will get the detailed figures to evaluate these criteria as a function of rows (one row corresponds to a spectrum for one integration). The flagged and unflagged data are shown in red and blue, respectively.

9.10 hsd_imaging

Profile Map

Figure 29 shows the top of the summary page. Three types of profile maps are available in the WebLog: 1) The simplified profile map of the combined image per spw at the top, 2) a simplified profile map per antenna, and 3) a detailed profile map. In the simplified profile map, the magenta lines indicate the atmospheric transmission at each frequency. One transmission profile is

plotted for each ASDM processed. To access the simplified profile map per antenna, click the corresponding “Spectral Window”. Each spectrum of the simplified profile maps (either 1. or 2.) corresponds to an averaged spectrum in an area of ⅛ of the image size (imsize), so that the total number of spectra in the profile map is 8 times 8. If the number of pixels (along x- or y-axis) is less than eight, it shows all spectrum per pixel. To see the detailed profile maps, click the icon with a symbol of polarization in the polarization column (see Figure 29). Each bin of the profile map is equivalent to a pixel, but with an interval of three cells. Due to the limitation of the allowed number of plots per page (max 5 x 5 plots per page), the rest of the plots are displayed in other pages.

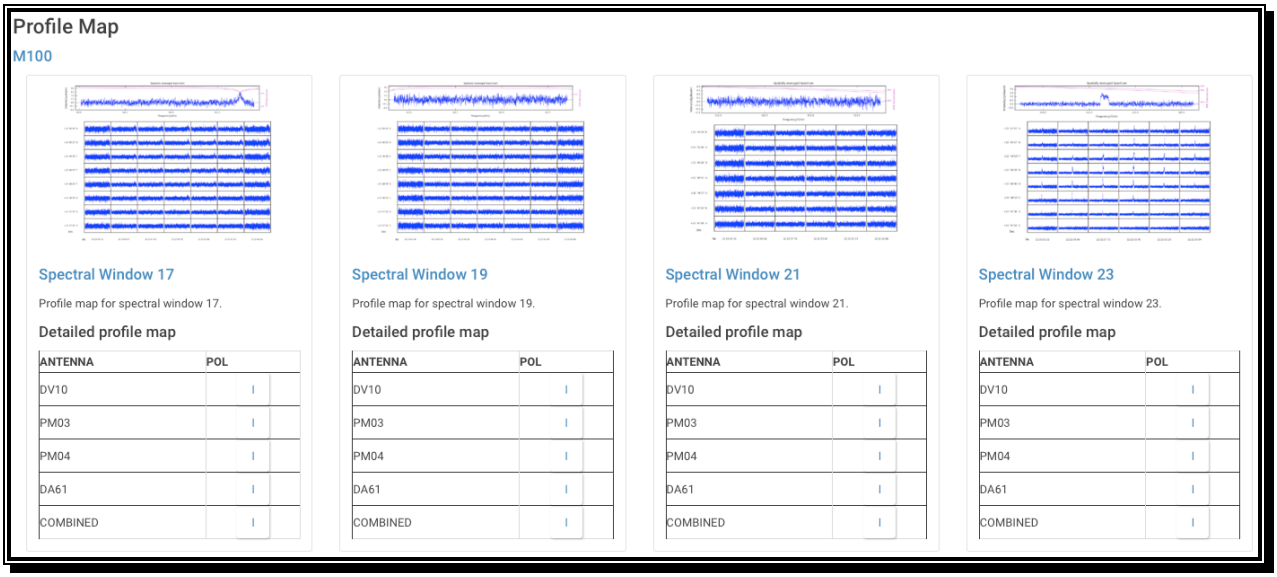


Figure 29 An example of the profile map.

Channel Map

The number of channel maps per spw corresponds to the number of emission lines that have been identified by the clustering analysis. In each channel map (see Figure 30), the top-middle plot shows the identified emission line and the determined line width (bracketed by two red vertical lines), overplotted on the averaged flux spectrum (in Jy) as a function of frequency (in GHz).

The top-left plot shows the zoom-up view of the identified emission line, but with velocity axis. The vertical axis is the averaged flux in Jy and the horizontal axis is in units of km/s. The (center) velocity of 0 km/s corresponds to the central frequency of the emission line, while the velocity range is equivalent to the masked region where the emission line was identified. The line velocity width is gridded into 15 bins, which are shown as red vertical lines. The top-right plot shows the total integrated intensity map (in Jy/beam km/s) over the all channels in the spw. Finally the channel maps within the velocity range of the identified emission line are shown in the panel at the bottom. Each channel plot corresponds to a bin in the top-left plot.

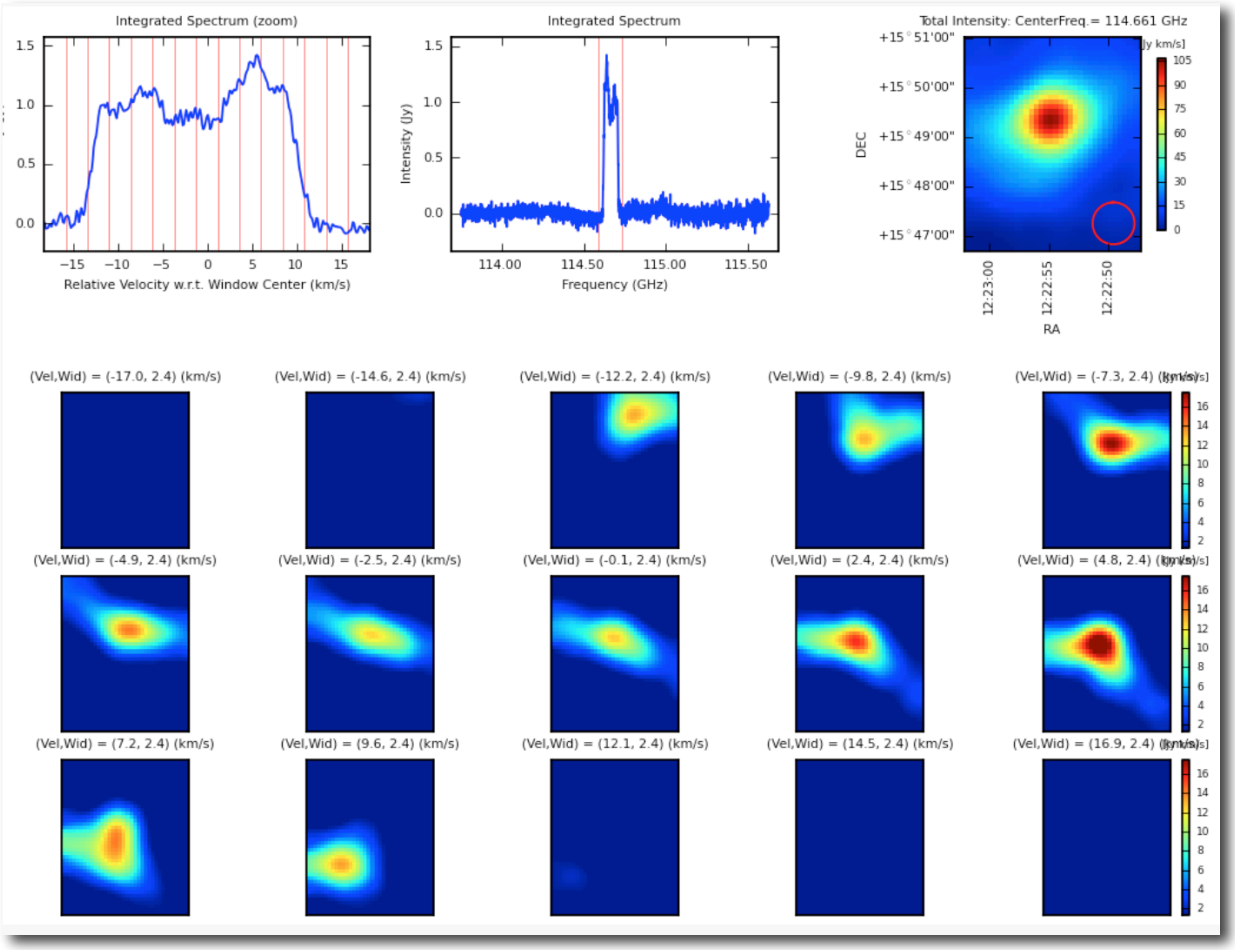


Figure 30 An example of channel map.

The **Baseline RMS Map** is created using the baseline RMS stored in the baseline tables. The baseline RMS is calculated by `hsd_baseline` using emission free channels.

The **Integrated Intensity Map** for each spw is generated using immoments task with all the available channel range.

The Atacama Large Millimeter/submillimeter Array (ALMA), an international astronomy facility, is a partnership of the European Organization for Astronomical Research in the Southern Hemisphere (ESO), the U.S. National Science Foundation (NSF) and the National Institutes of Natural Sciences (NINS) of Japan in cooperation with the Republic of Chile. ALMA is funded by ESO on behalf of its Member States, by NSF in cooperation with the National Research Council of Canada (NRC) and the National Science Council of Taiwan (NSC) and by NINS in cooperation with the Academia Sinica (AS) in Taiwan and the Korea Astronomy and Space Science Institute (KASI).

ALMA construction and operations are led by ESO on behalf of its Member States; by the National Radio Astronomy Observatory (NRAO), managed by Associated Universities, Inc. (AUI), on behalf of North America; and by the National Astronomical Observatory of Japan (NAOJ) on behalf of East Asia. The Joint ALMA Observatory (JAO) provides the unified leadership and management of the construction, commissioning and operation of ALMA.