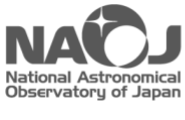
Doc 11.3, ver. 1.2 | Oct 2024

ALMA QA2 Data Products for Cycle 11



www.almascience.org

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

User Support:

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

Revision History:

Version	Date	Editors
0.1-1.0	January/February 2014	Dirk Petry
2.0, 2.1	September 2014	Dirk Petry
2.2	February 2015	Dirk Petry
3.0	November 2015	Dirk Petry
5.0	March 2018	Dirk Petry, Arielle Moullet, Mark Lacy
5.1	May 2018	Dirk Petry, Mark Lacy
7.12	February 2021	Luciano Cerrigone, Dirk Petry
8.2	February 2022	Fabrizia Guglielmetti
9.10	September 2022	Fabrizia Guglielmetti
10.2, 10.9	February/October 2023	Fabrizia Guglielmetti
11.2/11.3	February/October 2024	Fabrizia Guglielmetti et al.

Contributors

Fabrizia Guglielmetti (ESO), Dirk Petry (ESO), Kouichiro Nakanishi (NAOJ), Brian Mason (NRAO), Drew Brisbin (JAO), Luciano Cerrigone (JAO), Harold Francke (JAO), Misato Fukagawa (NAOJ), Andrés Guzmán (JAO), John Hibbard (NRAO), Erica Keller (NRAO), Mark Lacy (NRAO), Arielle Moulet (NRAO), Hiroshi Nagai (NAOJ), Theodoros Nakos (JAO), Tony Remijan (NRAO), Toshiki Saito (NAOJ), Thomas Stanke (ESO), Catarina Ubach (NRAO) and Martin Zwaan (ESO)

In publications, please refer to this document as:

Guglielmetti, F. et al., 2024, ALMA QA2 Data Products for Cycle 11, Version 11.3, ALMA

Table of contents

Acronyms

In the following is a list of acronyms frequently appearing in the text:

ALMA	Atacama Large Millimeter/Submillimeter Array
ARC	ALMA Regional Center
ASDM	ALMA Science Data Model
CASA	Common Astronomy Software Applications package
DP	Data Processing
EB	Execution Block
FOV	Field Of View
GOUS	Group Observing Unit Set
JAO	Joint ALMA Observatory
MOUS	Member Observing Unit Set
MS	Measurement Set
PB	Primary Beam
PI	Principal Investigator
QA	Quality Assurance
QA0	QA level 0
QA1	QA level 1
QA2	QA level 2
SB	Scheduling Block
TP	Total Power
UID	Unique Identification number

1 Introduction

The objective of ALMA Quality Assurance (QA) is to deliver reliable final data products to the Principal Investigators (PIs) that meet the control parameters outlined in the science goals. This ensures the data is calibrated to the required accuracy and contains no significant calibration or imaging artefacts. As with all previous cycles, imaging products for Cycle 11 will be produced on a "best effort" basis during the QA process.

The first step of QA (QA0) is conducted shortly after observations and is based on metadata and quick-look analysis at the Execution Block (EB) level. If QA0 results are marginal, additional procedures are employed to verify data quality. This extended process, named QA0+, involves executing a simplified calibration and imaging pipeline to provide a reliable proxy for the final image quality. During the first level of QA (QA1), the observatory monitors array and antenna performance parameters to ensure high data quality. The second level (QA2) focuses on data reduction, where data processing (DP) experts (i.e. Data Reducers and Data Reduction Managers) at JAO, the ARCs and/or the ARC-nodes inspect the products generated by the ALMA pipeline, either automatically or semi-interactively. This document outlines the final science data products that are archived and delivered to the PIs at the conclusion of the QA2 process.

The DP experts perform for each ALMA science dataset a detailed analysis, to confirm that the observations have achieved the science goals requested by the PI. In particular, frequency setup, spatial setup, angular resolution, and continuum and/or line detection sensitivity are verified.

If the requirements are met within the cycle-dependent tolerances (see chapter 11 of the ALMA Technical Handbook), the data are declared "QA2-pass", packaged in a standardized way, and delivered to the PI.

If, on the other hand, the requirements are not met, but additional observations of the Scheduling Block (SB) are no longer possible within the cycle time allotted to the project, the data are declared "QA2-semipass" and are also delivered soon after it has become clear that re-observation is not possible (the observing unit has "timed-out"). Data may also be delivered as QA2-semipass if it is determined that the data quality cannot be improved by further observations (for instance, in the case of dynamic-range limited noise[1]), or if all the datasets taken are classified as QA0-semipass and the project has timed out. In the latter case, only the raw data are delivered.

The delivery is made of two parts:

a) *the science products and supporting material,*

b) *the raw data.*

Part *b*, the raw data, consists of visibility datasets in ALMA native format, i.e. the ALMA Science Data Model (ASDM), which are needed if the PI would like to perform custom data calibration. Part *a* of the delivery, the "QA2 Products", consists of the QA2 data reduction scripts, log files, calibration and flagging tables, and the imaging products on which the quality assurance decision was based. This document describes the contents of the Products section of the delivery in detail and is aimed at ALMA users.

[1] Please note in case of ALMA Band 1 observations, data may exhibit a bandpass ripple that impacts spectral dynamic range. For more information, see the Knowledge Base article anticipated to be released in December 2024 at ALMA Science Helpdesk.

Until September 2014, QA2 work was entirely performed manually using the QA2 calibration script generator tool[2]. We refer to data processed in this way as "**manually calibrated"**. In September 2014, the first official version of the ALMA Science Pipeline became available and has since then been upgraded several times. It is now used to calibrate most of the data. We refer to data processed by the Science Pipeline as "**pipeline calibrated**". In addition to the content in the QA2 report, the user can tell whether the data are pipeline calibrated by the presence of a "*PPR*.xml" or a "*pprequest.xml" file in the *script* subdirectory of the delivery package. See Section 4 for more details.

At the beginning of Cycle 4, the Science Pipeline was enabled to also perform imaging. This increased the processing capacity significantly. We refer to data processed by the imaging section of the Science Pipeline as **"pipeline imaged".** If the data are imaged by a data reducer using CASA without the help of the pipeline (typically using templates or the imaging script generator tool[1]), we call them as **"manually imaged"**. The approval of the imaging products remains in the hands of the data reducers and Data Reduction Managers. Manual imaging can be necessary even when the pipeline was used for calibration, depending on the specifics of the dataset. Sometimes manual imaging is performed in addition to pipeline imaging, for example to show the PI that the science goal can be reached just by tweaking one of the imaging parameters. In this case, the QA2 report will mention the delivery of **supplemental manual imaging**. Starting in Cycle 10, for full polarization data sets, the pipeline performs the intensity and the full polarization calibrations, while the imaging part follows manual procedures. Another advancement is the introduction of automated self-calibration heuristics for single-field non-ephemeris targets. For more details, see Chapters 10 and 11 of the Cycle 11 ALMA Technical Handbook. Full polarization calibration and self-calibration implementations in the pipeline are described at **https://almascience.org/processing/science-pipeline**.

2 Group ObsUnitSet, Member ObsUnitSet, and Scheduling Block

In ALMA's data structure, the smallest scheduling entity processed and delivered individually is the Member Observing Unit Set (MOUS). Each MOUS is the collection of all executions of one particular SB. The MOUS, therefore, represents all executions of a specific SB, and QA2 is conducted on an individual MOUS basis. This practice continues in Cycle 11, where QA2 is carried out across all pass Execution Blocks (EBs) associated with a single SB, thus corresponding to one MOUS. Products for each MOUS are delivered with a unique label containing the MOUS UID, i.e. a hexadecimal identifier that provides access to the ALMA archive.
MOUSes are organized into Group Observation Unit Sets (GOUSes). A GOUS is the set of all MOUSs which are meant to be combined to obtain the final image products of a give science goal. The GOUS corresponds exactly to one scince goal. In Cycle 11, the following types of MOUSes may comprise a single GOUS:

[2] The calibration and imaging script generators are CASA scripts which create a template script for data reduction, which is then edited by the data reducer. See Petry, D. et al. 2014, Proc. SPIE, 9152, 91520J

a) The 12 m Array MOUS(s),

b) The 7 m Array MOUS,

c) The TP observation MOUS.

The delivery packaging results in three tar files named as follows:

<Project ID>_<MOUS_UID>_001_of_001.tar
<Project ID>_<MOUS_UID>_auxiliary.tar
member_<MOUS_UID>_README.txt.tar

The archive offers the possibility to download individual product files. The auxiliary tar file contains calibration tables, logs, scripts and quality assurance reports, including the weblog. The tar file marked with 001_of_001 contains the FITS product files of the targets and possibly the calibrators. The README.txt.tar file contains a description of the main actions, categories and files of the ALMA data at the MOUS level. Before Cycle 4, the README file was providing information on what is delivered and how to work with it. This information has been transferred to the QA2 Report since Cycle 5.

Optionally, if the user would like to recreate the calibrated data, the user will need to download also the uncalibrated raw data and these will appear as tar files named

<Project ID>_<EB_UID>_asdm.sdm.tar

where EB_UID indicates the unique ID that identifies the execution block, since one tar file will be available per execution. By untarring any of the tar files in a delivery package, the user will find that a directory tree is built as described in the next Section.

3 Overview

An ALMA data delivery in Cycle 11 consists of the seven items shown in the directory tree below. Items 1 to 6 are the main delivery package (*part a*), which should be downloaded first. Item 7 (*part b*) is only required if the users would like to work with the raw visibility data. Once unpacked, all data fall into a standardized directory structure:

```
|-- project_id/
|   |-- science_goal.ouss_id/
|   |   |-- group.ouss_id/
|   |   |   |-- member.ouss_id/
|   |   |   |   |-- README        (1)     (member. MOUS_UID.README.txt)
|   |   |   |   |-- product/      (2)
|   |   |   |   |-- calibration/  (3)
|   |   |   |   |-- qa/           (4)
|   |   |   |   |-- script/       (5)
|   |   |   |   |-- log/          (6)
|   |   |   |   |-- raw/          (7)     (only present when part b is unpacked)
```

The directory content is summarized in the following. When an item is marked as "best effort", this means that it is generated by the data reducers only if sufficient resources are available and it is compatible with the science goal.

1) README

A text file explaining where to find information about the QA2 results. Please note that the main source of information on the data processing is the "weblog" included in the *qa* subdirectory: for more details see Section 4), i.e., QA documentation in *qa* subdirectory.

2) Imaging Products (in FITS format) in *product* subdirectory

a) for pipeline imaged data:

A set of continuum and line images for all science spectral windows, covering as large as possible a part of the spectral range. This set is as complete as possible given the computing resources of the processing pipeline, i.e. for datasets with very high spatial and/or spectral resolution some parts of the set may be automatically omitted by the pipeline, to keep the size of the product package within reasonable limits. This process is referred to as the "mitigation" of data products. From Cycle 10, self-calibrated imaging products are characterized by **selfcal** extension, to be distinguished by the regular imaging products. The regular imaging products generated by the pipeline include the **regcal** extension. However, during the ingestion process into the science Archive, the **regcal** extension is omitted from the regular imaging product filenames to maintain consistency with filenames from previous cycles, where self-calibration was yet not included as part of the pipeline. Starting from Cycle 11, cube imaging products will be delivered with self-calibration applied whenever possible; if self-calibration cannot be applied, only the regcal products will be provided. For comprehensive information regarding the pipeline output, including detailed descriptions of its processes and functionalities, please refer to the official ALMA pipeline documentation. This resource can be accessed through the following link: **https://almascience.org/processing/science-pipeline**

b) for pipeline calibrated data:

Continuum images of the bandpass and phase calibrators and (when present) the check source(s), for each of the science spectral windows for interferometric data (12-m and 7-m arrays) only.

c) manually imaged line cubes:

For the representative spectral window: One spectral cube of "representative channels" for at least the representative target in the MOUS, made at the requested spectral resolution considering the bandwidth specified for sensitivity. If the achieved spatial resolution is higher than requested, tapering may be applied if needed, to reach the science goal. Coarse continuum subtraction is applied for sources with bright continuum.

— (best effort) For each source and each spectral window: cubes of all species specifically listed in the proposal, made at an angular resolution as close as possible to the requested one and including

line-free channels on either side. Continuum subtraction is applied for sources with bright continuum. A note may be added if additional species are found in the data, but imaging may not be carried out.

— (best effort) Continuum image of all non-edge, non-line channels.

d) manually imaged continuum and or aggregate bandwidth images:

— One image of all continuum spectral windows, all non-edge channels, for at least the representative target in the MOUS.

— (best effort) One image of all continuum spectral windows, all non-line channels.

3) CASA Tables (in CASA Table format) in *calibration* subdirectory

a) for pipeline calibrated data

- Calibration tables: including Tsys, WVR, Bandpass, Gain, Amplitude (*caltables.tgz)
- List of the applycal commands for the delivered calibration table (*calapply.txt)
- Flagversions tables (*flagversions.tgz)
- Auxilliary products (*auxproducts.tgz):
  - Calibrator fluxes from the ALMA calibration database (flux.csv)
  - Calibration table antenna positions (antennapos.csv)
  - Continuum frequency ranges (cont.dat)
  - Flagging commands added manually by the pipeline operator (*flagtemplate.txt, *flagtargetstemplate.txt, *flagtsystemplate.txt)
  - Pipeline *.selfcal.json restore file for self calibration
  - "Kelvin to Jansky" calibration factors for TP-array observations (jyperk_query.csv or jyperk.csv).

See **https://almascience.org/processing/science-pipeline** for more information on the role of these files.

b) for manually calibrated data

- Calibration tables: including Tsys, WVR, Bandpass, Gain, Amplitude
- Flagversions tables
- Calibration plots and wvrgcal diagnostic output.

4) QA documentation in *qa* subdirectory

a) for pipeline-calibrated and/or imaged data

— The Pipeline Weblog, i.e. a system of webpages containing all the diagnostic plots and other information generated by the pipeline. This is the main source of information on all properties of the data and the steps taken to calibrate/image and flag them. The weblog also contains the summary of the commands executed by the pipeline, the casa log file for each stage and its QA scoring.

— A QA0 Report for each EB UID in pdf format. This document is a summary of what occurred during data acquisition. It contains information about the system and receiver temperature in every antenna, the amount of time spent on calibrators and target sources, and a first quality evaluation of the data.

— A QA2 Report for the MOUS UID in pdf format. This report contains information about the quality assessment performed after data processing, as well as comments from the reviewer responsible for approving the dataset for delivery. Starting from Cycle 10, the observatory has implemented automatic delivery of MOUSes that meet quality assurance standards without any detected issues. In such cases, the reviewer comments are autogenerated, confirming that no problems were encountered during the calibration or imaging processes. The report also contains Instructions on the contents of the data package and guidelines for restoring the data.

b) for manually calibrated/imaged data

- QA Reports for each EB of the imaged data (png and txt files).
- The Manual Weblog is provided in html format. The Manual Weblog contains the information of the CASA reduction scripts and related images: CASA reduction scripts and diagnostic plots are displayed, fits images are listed. Scripts can be downloaded.
- If pipeline tasks were used in the manual imaging script, there may also be the "imaging_weblog.tgz" file. Unpack and open with a browser like the normal Pipeline Weblog mentioned under (a) above.

5) Data Reduction Scripts (ASCII files) in *script* subdirectory

a) for pipeline calibrated and imaged data

- The Python scripts needed to restore the calibrated MeasurementSet(s) (MS(s)) (*scriptForPI.py and *casa_piperestorescript.py) or rerun the entire pipeline (*casa_pipescript.py).
- The pipeline processing request file (PPR*.xml or *pprequest.xml).

b) for pipeline calibrated and manually imaged data

- (in addition to **a)**): The imaging preparation script (if necessary) and the imaging script.
- Optionally also the *scriptForPolCalibration.py

c) for manually calibrated and imaged data

— The Python scripts needed to restore the calibrated MS(s) (*scriptForPI.py, *scriptForCalibration.py, *scriptForSDCalibration.py and optionally scriptForImagingPrep.py, scriptForPolCalibration.py)

— Imaging script, and the script to set up and run the calibration.

6) calibration and imaging log files in *log* subdirectory

— CASA log files from the QA2 processing in case of manually calibrated or manually imaged datasets.

— CASA commands log from the QA2 processing in case of pipeline reduced datasets.

7) Datasets (in ASDM format) in *raw* subdirectory (if downloaded)

— (for optional download) The EBs that were used in the imaging/QA2 assessment.

— (for optional download if applicable) Additional datasets that were NOT used in the data reduction but may still contain valuable scientific information.

4 Data Delivery Products

4.1 README

The complete name of this file in normal deliveries is member.<MOUS_UID>.README.txt. Until the end of Cycle 4, this was a text file summarizing the QA2 results and explaining the structure of the tarball and file naming conventions. Since Cycle 5 this README file has normally been just a reference to where one can find additional information. Summary of the QA2 results, explanation of the tarball structure and file naming convention are provided by the QA2 report, located in the *qa* directory.

4.2 Data Reduction Scripts (*script* directory)

In the *script* subdirectory, the user finds the CASA data reduction scripts (Python) which were used to calibrate and image the data. Depending on whether the Science Pipeline was used or the data were calibrated manually, the content of this directory varies:

1) *uid....ms.scriptForCalibration.py*: one calibration script for each EB of manually calibrated interferometric data.
2) *uid....ms.scriptForSDCalibration.py*: one calibration script for each EB of manually calibrated single-dish data.
3) *scriptForImagingPrep.py* (for manually imaged datasets, if needed): this script contains all necessary steps to prepare imaging that needs to be performed on all calibrated EBs

together (and therefore cannot be performed by the calibration script for the individual EBs). These steps may include alignment of frequency grids before concatenation, the concatenation into one MS, adjustments to the flux calibration etc. The result is a MS or a set of MSs that are ready for the final imaging. If present, this script is run automatically as the last step by the *scriptForPI.py* (see below).

4) *scriptForPolCalibration.py* (for manually calibrated full-polarization data): special calibration routine for full-polarization data that is executed after the per-EB-calibration scripts. If present, it will be run automatically by *scriptForPI.py* (see below).

5) *scriptForImaging.py* (for interferometric data): this script generates the same imaging products (see Section 5 below) as those that are stored in the *product* subdirectory of the delivery. Where convenient for the data reducer, the *scriptForImaging.py* may also contain elements that could in principle be put into a *scriptForImagingPrep.py*. In that case, a *scriptForImagingPrep.py* is not created.

6) *scriptForSDImaging.py* (for manually imaged single-dish data): this script generates the same imaging products (see Section 5 below) as those that are stored in the *product* subdirectory of the delivery.

7) ***scriptForPI.py***: in all cases, whether pipeline or manually calibrated, run this script in the *script* directory in order to regenerate the calibrated MS of the delivered data, after downloading the raw ASDMs. The README file and the QA2 report contain instructions (or point to them) on how to do this. Running the *scriptForPI.py* requires that you have downloaded the ASDM datasets and unpacked them, so that they reside in the *raw* directory (created during unpacking). See next Section for more details.

   Until mid-Cycle 5, the only type of dataset that was *not* delivered with a *scriptForPI.py* was the *manually calibrated TP* data. Since mid-Cycle 5, all datasets are delivered with scriptForPI.py. See Section 5 below.

8) *casa_piperestorescript.py*: this script is used by *scriptForPI.py* in case of pipeline-calibrated data to restore the calibrated MS(s) from the raw ASDMs by applying the necessary calibration tables. If it is not provided or it has been removed, the *scriptForPI.py* will run the *casa_pipescript.py* instead.

9) *casa_pipescript.py*: this script performs the calibration from scratch using Pipeline tasks. See the ALMA Science Pipeline User's Guide documentation for more details.

10) The pipeline processing request (PPR): an XML file that was used to drive the Pipeline calibration and/or imaging. See **https://almascience.org/processing/science-pipeline**

A script named *scriptForFluxCalibration.py* may be present in the *script* directory and executed by *scriptForPI.py* before *scriptForImagingPrep.py*. During Cycle 7, this script is employed with CASA 6.1 to evaluate the requirement of rescaling due to any lines detected and to apply a flux offset correction when needed. This renormalization process is evaluated by the pipeline from CASA 6.2 (see Section 5.7 below). Furthermore, there may be the following script file present in case of solar observations: *sun_reduction_util.py* (contains helper routines specific to solar data analysis). In

addition to the scripts, *manifest.xml and *pldriver_report.xml are products of the pipeline execution, containing information about the packaged files and metadata, as well as the list of associated EBs.

4.3 Imaging Products (*product* directory)

The *product* subdirectory of the delivery contains the imaging products that were generated either by the imaging section of the Science Pipeline or by a data reducer with the script scriptForImaging.py. These imaging products were created from the calibrated dataset produced when the scriptForPI.py is run. For pipeline calibrated interferometric data, this directory also contains calibrator images generated by the Pipeline.
A coherent **file naming scheme** was introduced for all imaging products:
<MOUS>.<target>_<target type>.spw<SPWs>.<imagetype>.<y>.<stokes>.<comment>.<x>.<ext>
Where

<MOUS>	Is the MOUS UID (for example, uid___A001_X23a_X2c).
<target>	Is the name of the imaged field (it can be the name of a calibrator or of the scientific target).
<target type>	Is "sci" for science target, "ph" for phase calibrator, "bp" for bandpass calibrator, "amp" for flux calibrator (this will not be present if the bandpass calibrator is also used as flux calibrator), "chk" for check source, "pol_leak" for the polarization calibrator.
<SPWs>	Are the imaged spectral window IDs.
<image type>	Is "mfs" for aggregate bandwidth image, "cube" for a spectrally resolved image, "repBW" for cube at representative bandwidth, "cont" for an image with the line channels excluded (in pipeline calibrated/imaged data, this will have all spectral windows merged). For multi-term wideband imaging, additional .tt0,.tt1, etc suffixes indicate Taylor terms, plus .alpha for spectral index image, .alpha.error for error on spectral index estimate (error map), .beta for spectral curvature if nterms > 2.
<y>	From Cycle 10, the image type as outcome from regular or self calibration (y=='selfcal' for self-calibrated images, while y is blank for regular calibration images).
<stokes>	The name of the image Stokes parameter (e.g., "I").
<comment>	Can be an explanatory text, such as a molecular transition.
<x>	Is "pbcor" for the primary beam corrected image, "pb" for the PB itself, "mask" for the mask image, "sd" for single dish.
<ext>	Is the filename extension describing the file type, e.g. "fits".

Note that the ***<comment>*** field typically contains the word “manual” if the image was obtained by manual imaging. Information about the ALMA naming scheme can also be found in the following knowledgebase article

https://help.almascience.org/kb/articles/what-calibration-and-imaging-products-will-be-delivered-to-me

The goal of ALMA QA2 in Cycle 11 remains that the calibration is reliable and "science ready", but the imaging products may not be. A sufficient fraction of the possible imaging products are created to verify that the data meet the science goals set by the PI (resolution and sensitivity), within some tolerance, but the **imaging may be incomplete**. In general, investigators should expect to need to re-image their data using the provided scripts (for manual imaging) or the pipeline documentation (for pipeline imaged data) as a guideline. In particular, care should be taken to optimize continuum subtraction (if relevant) and fine-tune the image deconvolution parameters (e.g. clean masks and thresholds). See “ALMA Imaging Pipeline Reprocessing” in CASA guides:

https://casaguides.nrao.edu/index.php?title=ALMA_Imaging_Pipeline_Reprocessing

Images conform to the FITS standard 3.0. The imaging clean masks are provided as gzipped FITS files. These represent the final mask used in the last cleaning iteration. The masks used in previous iterations may have been different (smaller in area).

The science images included in deliveries of interferometric data are corrected for the primary beam (PB), i.e. the dependence of the instrument sensitivity on direction within the field of view (FOV). For each image, two files are delivered:

a) The PB-corrected image (file name ending in ".pbcor.fits").

b) The image of the PB used for the correction ("*.pb.fits" or “*.flux.fits”, possibly gzipped).

The image noise is measured in the uncorrected image. The corrected image (a) is then obtained by dividing the uncorrected image by the PB image (b). The uncorrected image can be recovered using the CASA task “impbcor” in mode "m":

```
impbcor(imagename='image.pbcor.fits',pbimage='image.flux.fits', mode='m',
outfile='image.recovered')
```

For **TP** data, images are delivered accounting for the convolution of the sky intensity distribution with the PB of the antenna, implying no need for a PB correction. In early cycles and high frequency regular observations, additional amplitude calibrator observation was taken whose images were not included.

If the data reducer deemed it necessary, in the case of pipeline imaged data, **supplemental manual imaging** may be added to the delivery package. The user will then find in the *product* directory a file named *manual_imaging.tgz* containing the CASA log files and the scripForImaging.py used to perform such supplemental imaging. The user should take care not to mix up this scriptForImaging.py with that in the *script* directory, which is related to the pipeline run. The supplemental products will contain the comment *manual* in their names and can be found either inside the *manual_imaging.tgz* or simply in the *products* directory along with the pipeline products.

4.4 Calibration Tables (*calibration* directory)

In order to further document the process of data calibration, all calibration tables generated by the calibration process are stored in the *calibration* subdirectory of the delivery.
In the case of manually calibrated data, information on how each table was created can be found in the scriptForCalibration.py and the scriptForImagingPrep.py scripts. CASA provides functionality to view these tables. Please refer to the CASA documentation (see section 6, in particular the ALMA Science Pipeline User's Guide) on how to do that. However, many of the plots of interest in this context are also available in the QA documentation (see Section 4.5 below).

In the case of pipeline-calibrated data, the files in this directory are needed to restore the calibrated MS with "*casa_piperestorescript.py*" which is called by "*scriptForPI.py*". For details on how they were generated, please refer to **https://almascience.org/processing/science-pipeline**. All diagnostic plots are also available in the Pipeline Weblog, which can be found in the *qa* directory (see Section 4.5 below).

4.5 QA Documentation (*qa* directory)

In the *qa* subdirectory of the delivery, the user finds diagnostic information obtained during calibration and imaging.

In the case of *pipeline calibrated data,* all diagnostic information is assembled in a system of html pages which is called the **Weblog**. To access the Weblog, run

```
tar xzf *hifa*weblog.tar.gz
```

in the *qa* directory and then navigate to the root of the untarred weblog directory e.g. pipeline-20220914T170517 and run h_weblog() within a CASA session containing pipeline tasks. It is recommended to set your default browser to Firefox to view the weblog, although since the 2021 pipeline release there has been limited support for Chrome and Safari; for more information refer to **https://help.almascience.org/kb/articles/what-is-the-best-way-to-view-the-weblog**.
For more information on the contents of the Weblog, please refer to the Pipeline documentation at **https://almascience.org/processing/science-pipeline**.

In the case of *manually calibrated* data, there are several png images and a textfile for each EB, labelled with the EB UID. The png files can be viewed with standard system tools such as "xdg-open" (under Linux) or with the command "open" under MacOS. They show a set of diagnostic plots describing the following aspects of the data:

1) Observing Schedule (observation intent vs. time).
2) Mosaic Pointing Configuration.
3) Antenna Array Configuration.
4) Effectiveness of the WVR correction for each antenna (for 12M observations).
5) Temporal gain calibration solutions for each antenna.
6) Temporal phase calibration solutions for each antenna.
7) Average bandpass solution for each spectral window.
8) System Temperature vs. frequency for each antenna and each spectral window.

9) Phase calibrator amplitude and phase vs. frequency for each spectral window and polarization.
10) Flux calibration model and data (visibility amplitude vs. UV distance).
11) Target visibility amplitude vs. UV distance for each spectral window and polarization.
12) Phase calibrator amplitude and phase vs. frequency for each spectral window and polarization.
13) Target field UV coverage.
14) Test image of the target.
15) Target imaging synthesized beam (PSF).
16) Phase stability plot (PhaseRMS vs. baseline length).

The content of the uid*textfile.txt file is mostly self-explanatory. Of particular interest is the "Check of a target image and sensitivity", which mentions approximately the achieved spatial resolution and sensitivity. Note that the finally achieved resolution and sensitivity could be different, due to additional procedures in imaging.

From Cycle 7, all available png images and scripts related to the manual data reduction are assembled in the manual Weblog, a system of html pages. To access the Weblog, run

```
tar xzf *manual*weblog.tar.gz
```

in the *qa* directory and then use your favourite web browser to open the resulting file:

```
manual*/html/index.html
```

In case of manual calibration, all EBs are expandable links with the related diagnostic information. In case of manual imaging procedure, the imaging tab provides the related imaging scripts and list of plots. Each script is equipped with a link to visualize the content.

Based on the calibration tables in the *calibration* directory, the user can produce additional diagnostic plots in CASA using the tasks *plotms* and *plotbandpass*. See the CASA documentation for more details.

If additional diagnostic plots on the visibility data themselves are required, one needs first to generate the calibrated MS from the raw data (see Section 5 below). Once this is done, the CASA task *plotms* can be used to generate the plots.

5 Generating the calibrated visibilities

In ALMA Cycle 0 and early Cycle 1, the visibility data were delivered with all other products in ready-to-use MS format. Since the middle of Cycle 1, the products are separated from the raw visibility data (as described in the introduction); downloads of the latter are now optional. To minimize download time, the raw visibilities are delivered in the native ALMA format, the ASDM.

If the user would like to only modify or extend the imaging of the dataset, but accepts the observatory calibration of the data, they need to download the raw data and apply the calibration via the scripts provided with the delivery as described in the following. Otherwise, the user can download the raw data and use the calibration scripts or pipeline documentation as a guidance to develop a custom calibration. Moreover, each ARC provides calibrated Measurement Sets directly:

https://help.almascience.org/kb/articles/how-do-i-obtain-a-file-of-calibrated-visibilities-measurement-set-for-alma-data

The download of the raw data depends on the details of where the data were staged for delivery. The user should refer to the information in the delivery email as to how exactly the raw data should be obtained. Unless there is a technical problem, this will work via the ALMA Archive Request Handler. See **http://almascience.org/alma-data**.

5.1 Running the *scriptForPI.py*

Both for manually and pipeline-calibrated data, the calibrated visibilities are restored by unpacking the tarballs obtained from the Archive into a designated location, navigating to the "script" directory, launching the appropriate CASA version, and running <member...scriptForPI.py> via the execfile command. This process is described in detail in the following sections.

5.2 Running the *scriptForPI.py* in the case of manually calibrated data

For manually calibrated data, the scriptForPI.py will find the scripts named scriptForCalibration.py and, e.g., scriptForImagingPrep.py (if present), set up every necessary item for their execution, and then run them to obtain the MS ready for imaging.

Once the data-products tar file is downloaded and unpacked, the user finds a certain directory structure on disk (see the description in Section 3). The following directory tree shows an example of manually calibrated (rather than pipeline calibrated) data. However, also for pipeline calibrated data, the procedure for running "*scriptForPI.py*" is the same and has essentially the same final product, although the naming differs.

```
2023.1.01573.S
└── science_goal.uid___A001_X3621_X53f
   └── group.uid___A001_X3621_X540
      └── member.uid___A001_X3621_X541
         ├── calibration
         |   ├── uid___A002_X110fd13_X12b88.calibration.plots.tgz
         |   ├── uid___A002_X110fd13_X12b88.calibration.tgz
         |   ├── uid___A002_X110fd13_X12b88.ms.wvrgcal.txt
         |   ├── uid___A002_X110fd13_X131c8.calibration.plots.tgz
         |   ├── uid___A002_X110fd13_X131c8.calibration.tgz
         |   ├── uid___A002_X110fd13_X131c8.ms.wvrgcal.txt
         |   ├── uid___A002_X110fd13_X1b52b.calibration.plots.tgz
         |   ├── uid___A002_X110fd13_X1b52b.calibration.tgz
         |   ├── uid___A002_X110fd13_X1b52b.ms.wvrgcal.txt
         |   ├── uid___A002_X110fd13_X230eb.calibration.plots.tgz
         |   ├── uid___A002_X110fd13_X230eb.calibration.tgz
         |   ├──uid___A002_X110fd13_X230eb.ms.wvrgcal.txt
```

```
|   ├── uid___A002_X110fd13_X59fd.calibration.plots.tgz
|   ├── uid___A002_X110fd13_X59fd.calibration.tgz
|   └── uid___A002_X110fd13_X59fd.ms.wvrgcal.txt
├── log
|   ├── member.uid___A001_X3621_X541*.log.tgz
|   ├── uid___A002_X110fd13_X12b88.log.tgz
|   ├── uid___A002_X110fd13_X131c8.log.tgz
|   ├── uid___A002_X110fd13_X1b52b.log.tgz
|   ├── uid___A002_X110fd13_X230eb.log.tgz
|   └── uid___A002_X110fd13_X59fd.log.tgz
├── member.uid___A001_X3621_X541.README.txt
├── product
|   ├── member..J2206-0031_sci.spw25.cube.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw25.cube.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw25.cube.I.pbcor.fits
|   ├── member..J2206-0031_sci.spw25.mfs.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw25.mfs.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw25.mfs.I.pbcor.fits
|   ├── member..J2206-0031_sci.spw25_27_29_31_33.cont.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw25_27_29_31_33.cont.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw25_27_29_31_33.cont.I.pbcor.fits
|   ├── member..J2206-0031_sci.spw27.cube.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw27.cube.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw27.cube.I.pbcor.fits
|   ├── member..J2206-0031_sci.spw27.mfs.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw27.mfs.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw27.mfs.I.pbcor.fits
|   ├── member..J2206-0031_sci.spw29.cube.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw29.cube.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw29.cube.I.pbcor.fits
|   ├── member..J2206-0031_sci.spw29.mfs.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw29.mfs.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw29.mfs.I.pbcor.fits
|   ├── member..J2206-0031_sci.spw31.cube.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw31.cube.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw31.cube.I.pbcor.fits
|   ├── member..J2206-0031_sci.spw31.mfs.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw31.mfs.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw31.mfs.I.pbcor.fits
|   ├── member..J2206-0031_sci.spw33.cube.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw33.cube.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw33.cube.I.pbcor.fits
```

```
|   ├── member..J2206-0031_sci.spw33.mfs.I.mask.fits.gz
|   ├── member..J2206-0031_sci.spw33.mfs.I.pb.fits.gz
|   ├── member..J2206-0031_sci.spw33.mfs.I.pbcor.fits
|   ├── member..J2136p0041_chk.spw25.mfs.I.mask.fits.gz
|   ├── member..J2136p0041_chk.spw25.mfs.I.pb.fits.gz
|   ├── member..J2136p0041_chk.spw25.mfs.I.pbcor.fits
|   ├── member..J2136p0041_chk.spw27.mfs.I.mask.fits.gz
|   ├── member..J2136p0041_chk.spw27.mfs.I.pb.fits.gz
|   ├── member..J2136p0041_chk.spw27.mfs.I.pbcor.fits
|   ├── member..J2136p0041_chk.spw29.mfs.I.mask.fits.gz
|   ├── member..J2136p0041_chk.spw29.mfs.I.pb.fits.gz
|   ├── member..J2136p0041_chk.spw29.mfs.I.pbcor.fits
|   ├── member..J2136p0041_chk.spw31.mfs.I.mask.fits.gz
|   ├── member..J2136p0041_chk.spw31.mfs.I.pb.fits.gz
|   ├── member..J2136p0041_chk.spw31.mfs.I.pbcor.fits
|   ├── member..J2136p0041_chk.spw33.mfs.I.mask.fits.gz
|   ├── member..J2136p0041_chk.spw33.mfs.I.pb.fits.gz
|   └── member..J2136p0041_chk.spw33.mfs.I.pbcor.fits
├── qa
|   ├── member.uid___A001_X3621_X541.imaging_weblog.tgz
|   ├── member.uid___A001_X3621_X541.manual_calimage.weblog.tgz
|   ├── member.uid___A001_X3621_X541.qa2_report.pdf
|   ├── uid___A002_X10f72bf_X1383.qa0_report.pdf
|   ├── uid___A002_X110fd13_X12b88.qa0_report.pdf
|   ├── uid___A002_X110fd13_X12b88__qa2_part*.png
|   ├── uid___A002_X110fd13_X12b88__textfile.txt
|   ├── uid___A002_X110fd13_X131c8.qa0_report.pdf
|   ├── uid___A002_X110fd13_X131c8__qa2_part*.png
|   ├── uid___A002_X110fd13_X131c8__textfile.txt
|   ├── uid___A002_X110fd13_X1b52b.qa0_report.pdf
|   ├── uid___A002_X110fd13_X1b52b__qa2_part*.png
|   ├── uid___A002_X110fd13_X1b52b__textfile.txt
|   ├── uid___A002_X110fd13_X230eb.qa0_report.pdf
|   ├── uid___A002_X110fd13_X230eb__qa2_part*.png
|   ├── uid___A002_X110fd13_X230eb__textfile.txt
|   ├── uid___A002_X110fd13_X59fd.qa0_report.pdf
|   ├── uid___A002_X110fd13_X59fd__qa2_part*.png
|   └── uid___A002_X110fd13_X59fd__textfile.txt
└── script
    ├── member.uid___A001_X3621_X541.calimage.product_rename.txt
    ├── member.uid___A001_X3621_X541.scriptForImaging.py
    ├── member.uid___A001_X3621_X541.scriptForPI.py
```

├── *uid___A002_X110fd13_X12b88.ms.**scriptForCalibration.py***
├── *uid___A002_X110fd13_X12b88__calibratorFluxes.xml*
├── *uid___A002_X110fd13_X131c8.ms.scriptForCalibration.py*
├── *uid___A002_X110fd13_X131c8__calibratorFluxes.xml*
├── *uid___A002_X110fd13_X1b52b.ms.scriptForCalibration.py*
├── *uid___A002_X110fd13_X1b52b__calibratorFluxes.xml*
├── *uid___A002_X110fd13_X230eb.ms.scriptForCalibration.py*
├── *uid___A002_X110fd13_X230eb__calibratorFluxes.xml*
├── *uid___A002_X110fd13_X59fd.ms.scriptForCalibration.py*
└── *uid___A002_X110fd13_X59fd__calibratorFluxes.xml*

Note that for simplicity, the names of the files in the *product* directory were shortened here by removing the MOUS UID, which in a real delivery would be contained within the double dots in the names of our example. In the example above, the *qa* directory contains a qa2 report in pdf format. The html format of the qa2 report has been deprecated during Cycle 7. From Cycle 8, the manual weblog is provided in html format and compressed in a tgz file. Because pipeline imaging tasks were used during the imaging process, an * imaging_weblog.tgz is present in the *qa* directory.

This dataset comes from a bandwidth-switching observation with five EBs, each assigned a unique UID. The corresponding calibration scripts in the script directory are named to match each UID:

├── uid___A002_X110fd13_X12b88.ms.scriptForCalibration.py
├── uid___A002_X110fd13_X131c8.ms.scriptForCalibration.py
├── uid___A002_X110fd13_X1b52b.ms.scriptForCalibration.py
├── uid___A002_X110fd13_X230eb.ms.scriptForCalibration.py
└── uid___A002_X110fd13_X59fd.ms.scriptForCalibration.py

In case of full polarization manual calibration, a script for this aim is additionally present and named *.*scriptForPolCalibration.py*

Furthermore, like in every delivery, the *script* directory contains the scripts "**scriptForImaging.py*" and "**scriptForPI.py*".

├── member.uid___A001_X3621_X541.scriptForImaging.py
├── *member.uid___A001_X3621_X541.scriptForPI.py*

In case of polarization manual data reduction, the user could have also found a *scriptForImagingPrep.py*, which is an optional script sometimes used by a data reducer to perform further adjustments (if deemed necessary) after the calibration, for example flux equalization.

The user is encouraged to inspect all these scripts, as they may also contain helpful comments on special properties of the data.

Unpacking the raw data after downloading should be done at the top level (in the directory containing the directory 2023.1.01573.S in this example). It should result in an additional *raw* directory at the level of the *script* directory:

```
|   |   |   |   |-- raw/
|   |   |   |   |   |-- uid___A002_X110fd13_X12b88.asdm.sdm
|   |   |   |   |   |-- uid___A002_X110fd13_X131c8.asdm.sdm
|   |   |   |   |   |-- uid___A002_X110fd13_X1b52b.asdm.sdm
|   |   |   |   |   |-- uid___A002_X110fd13_X230eb.asdm.sdm
|   |   |   |   |   |-- uid___A002_X110fd13_X59fd.asdm.sdm
```

In order to just reproduce the observatory calibration, the user can then run the "*scriptForPI.py*" by typing at the shell prompt of the operating system:

```
cd script
casa -c "execfile('member.uid___A001_X3621_X541.scriptForPI.py')"
```

Note that the same version of CASA should be used as that of the original processing.

Executing scriptForPI.py will first perform various tests on the directory structure and the presence of the necessary files and then run the corresponding scriptForCalibration.py on each of the ASDMs. If there is more than one ASDM and further steps were necessary in the QA2 process to prepare the dataset for imaging, the scriptForImagingPrep.py will be present and will be run as well.
The results will be placed in a new directory called *calibrated* at the same level as the directories "raw" and "script":

```
|   |   |   |   |-- calibrated
|   |   |   |   |   |-- calibrated.ms
|   |   |   |   |   |-- uid___A002_ X110fd13_X12b88.calibration
|   |   |   |   |   |-- uid___A002_ X110fd13_X12b88.ms.split.cal
|   |   |   |   |   |-- uid___A002_ X110fd13_X131c8.calibration
|   |   |   |   |   |-- uid___A002_ X110fd13_X131c8.ms.split.cal
|   |   |   |   |   |-- uid___A002_ X110fd13_X1b52b.calibration
|   |   |   |   |   |-- uid___A002_ X110fd13_X1b52b.ms.split.cal
|   |   |   |   |   |-- uid___A002_ X110fd13_X230eb.calibration
|   |   |   |   |   |-- uid___A002_ X110fd13_X230eb.ms.split.cal
|   |   |   |   |   |-- uid___A002_X110fd13_X59fd.calibration
|   |   |   |   |   |-- uid___A002_X110fd13_X59fd.ms.split.cal
```

If the data reducer deems it necessary, the final calibrated visibility data combined for all EBs can be found in a MS called "calibrated.ms". The concatenated MS can also not be present, if the data reducer does not deem it necessary to perform a concatenation, and the imaging will operate directly on the MSs for each EB. The intermediate output of CASA to achieve the calibration of each EB is stored in the calibration working directories:

```
|   |   |   |   |   |-- uid___A002_ X110fd13_X12b88.calibration
|   |   |   |   |   |-- uid___A002_ X110fd13_X131c8.calibration
|   |   |   |   |   |-- uid___A002_ X110fd13_X1b52b.calibration
|   |   |   |   |   |-- uid___A002_ X110fd13_X230eb.calibration
|   |   |   |   |   |-- uid___A002_X110fd13_X59fd.calibration
```

while the calibrated visibilities for each EB individually are stored in the MSs with names ending in ".ms.split.cal" (if there is only one EB, this MS will take the role of calibrated.ms).

The execution of scriptForPI.py will produce MSs that are ready for imaging, but will not perform any imaging on them: it will not run scriptForImaging.py.

5.3 Running the *scriptForPI.py* in the case of pipeline-calibrated data

The following shows the contents of the directory tree for a dataset that was calibrated and imaged using the ALMA pipeline. This can easily be seen from the fact that there is a file "***pprequest.xml**" present in the *script* directory. Like in the case of manually calibrated data, the "*scriptForPI.py*" is found in the *script* directory.

2024.1.00657.S
└── science_goal.uid___A001_X3788_Xc375
  └── group.uid___A001_X3788_Xc37c
    └── member.uid___A001_X3788_Xc37d
      ├── calibration
      | |-- member..hifa_calimage*.auxproducts.tgz
      | |-- member.uid___A001_X3788_Xc37d.s1.caltables.tgz
      | |-- uid___A002_X11e94c7_X4f7.ms.calapply.txt
      | |-- uid___A002_X11e94c7_X4f7.ms.flagversions.tgz
      ├── product
      | |-- member…_SAX_J1818.6-1703__**sci**.spw11.cube.I.selfcal.pb.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw11.cube.I.**selfcal.pbcor.fits**
      | |-- member…_SAX_J1818.6-1703__sci.spw11.mfs.I.mask.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw11.mfs.I.pb.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw11.mfs.I.pbcor.fits
      | |-- member…_SAX_J1818.6-1703__sci.spw11.mfs.I.selfcal.mask.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw11.mfs.I.selfcal.pb.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw11.mfs.I.selfcal.pbcor.fits
      | |-- member…_SAX_J1818.6-1703__sci.spw5.cube.I.selfcal.mask.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw5.cube.I.selfcal.pb.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw5.cube.I.selfcal.pbcor.fits
      | |-- member…_SAX_J1818.6-1703__sci.spw5.mfs.I.mask.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw5.mfs.I.pb.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw5.mfs.I.**pbcor.fits**
      | |-- member…_SAX_J1818.6-1703__sci.spw5.mfs.I.selfcal.mask.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw5.mfs.I.selfcal.pb.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw5.mfs.I.selfcal.pbcor.fits
      | |-- member..._SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.alpha.error.fits
      | |-- member…_SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.alpha.fits
      | |-- member…_SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.mask.fits.gz
      | |-- member…_SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.pb.tt0.fits

| |-- member…_SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.selfcal.alpha.error.fits
| |-- member…_SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.selfcal.alpha.fits
| |-- member…_SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.selfcal.mask.fits.gz
| |-- member…_SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.selfcal.pb.tt0.fits
| |-- member…_SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.selfcal.tt0.pbcor.fits
| |-- member..._SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.selfcal.tt1.pbcor.fits
| |-- member…_SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.tt0.pbcor.fits
| |-- member…_SAX_J1818.6-1703__sci.spw5_7_9_11.cont.I.tt1.pbcor.fits
| |-- member…_SAX_J1818.6-1703__sci.spw7.cube.I.selfcal.pb.fits.gz
| |-- member…_SAX_J1818.6-1703__sci.spw7.cube.I.selfcal.pbcor.fits
| |-- member…_SAX_J1818.6-1703__sci.spw7.mfs.I.mask.fits.gz
| |-- member…_SAX_J1818.6-1703__sci.spw7.mfs.I.pb.fits.gz
| |-- member…_SAX_J1818.6-1703__sci.spw7.mfs.I.pbcor.fits
| |-- member… SAX_J1818.6-1703__sci.spw7.mfs.I.selfcal.mask.fits.gz
| |-- member…_SAX_J1818.6-1703__sci.spw7.mfs.I.selfcal.pb.fits.gz
| |-- member… SAX_J1818.6-1703__sci.spw7.mfs.I.selfcal.pbcor.fits
| |-- member..._SAX_J1818.6-1703__sci.spw9.cube.I.selfcal.pb.fits.gz
| |-- member..._SAX_J1818.6-1703__sci.spw9.cube.I.selfcal.pbcor.fits
| |-- member..._SAX_J1818.6-1703__sci.spw9.mfs.I.mask.fits.gz
| |-- member..._SAX_J1818.6-1703__sci.spw9.mfs.I.pb.fits.gz
| |-- member..._SAX_J1818.6-1703__sci.spw9.mfs.I.pbcor.fits
| |-- member..._SAX_J1818.6-1703__sci.spw9.mfs.I.selfcal.mask.fits.gz
| |-- member..._SAX_J1818.6-1703__sci.spw9.mfs.I.selfcal.pb.fits.gz
| |-- member..._SAX_J1818.6-1703__sci.spw9.mfs.I.selfcal.pbcor.fits
| |-- member.uid___A001_X3788_Xc37d.J1832-2039_ph.spw11.mfs.I.mask.fits.gz
| |-- member...J1832-2039_**ph**.spw11.mfs.I.pb.fits.gz
| |-- member...J1832-2039_ph.spw11.mfs.I.pbcor.fits
| |-- member...J1832-2039_ph.spw5.mfs.I.mask.fits.gz
| |-- member...J1832-2039_ph.spw5.mfs.I.pb.fits.gz
| |-- member...J1832-2039_ph.spw5.mfs.I.pbcor.fits
| |-- member...J1832-2039_ph.spw7.mfs.I.mask.fits.gz
| |-- member...J1832-2039_ph.spw7.mfs.I.pb.fits.gz
| |-- member...J1832-2039_ph.spw7.mfs.I.pbcor.fits
| |-- member...J1832-2039_ph.spw9.mfs.I.mask.fits.gz
| |-- member...J1832-2039_ph.spw9.mfs.I.pb.fits.gz
| |-- member...J1832-2039_ph.spw9.mfs.I.pbcor.fits
| |-- member...J1924-2914_**bp**.spw11.mfs.I.mask.fits.gz
| |-- member...J1924-2914_bp.spw11.mfs.I.pb.fits.gz
| |-- member...J1924-2914_bp.spw11.mfs.I.pbcor.fits
| |-- member...J1924-2914_bp.spw5.mfs.I.mask.fits.gz
| |-- member...J1924-2914_bp.spw5.mfs.I.pb.fits.gz
| |-- member...J1924-2914_bp.spw5.mfs.I.pbcor.fits
| |-- member...J1924-2914_bp.spw7.mfs.I.mask.fits.gz
| |-- member...J1924-2914_bp.spw7.mfs.I.pb.fits.gz
| |-- member...J1924-2914_bp.spw7.mfs.I.pbcor.fits

```
|   |-- member...J1924-2914_bp.spw9.mfs.I.mask.fits.gz
|   |-- member...J1924-2914_bp.spw9.mfs.I.pb.fits.gz
|   |-- member...J1924-2914_bp.spw9.mfs.I.pbcor.fits
 ├── qa
|   |--  member..hifa_calimage*.weblog.tgz
|   |--  member..qa2_report.pdf
|   |--  uid___A002_X11e94c7_X4f7.qa0_report.pdf
 ├── raw
|   |--   uid___A002_X11e94c7_X4f7.asdm.sdm
└── script
|-- member...calimage.product_rename.txt
|-- member...hifa_calimage.casa_piperestorescript.py
|-- member...hifa_calimage.casa_pipescript.py
|-- member...hifa_calimage.pipeline_manifest.xml
|-- member...hifa_calimage.pldriver_report.xml
|-- member...hifa_calimage.pprequest.xml
`-- member...scriptForPI.py
```

The file names in the directories “product”, “qa”, and “script” were shortened by not showing the MOUS UID, which is indicated by two consecutive dots in the file name. Note the additional calibrator images created by the Science Pipeline.

As with a manually calibrated dataset, if you want to perform further imaging or re-calibrate, you will first need to download and unpack the raw data. This will create the additional *raw* subdirectory containing the ASDM(s). In order to then run “*scriptForPI.py*”, you will need to verify that you have installed the CASA version indicated in the QA2 Report or Weblog. Then start up CASA in the working *script* directory using

```
casa --pipeline
```

At the CASA prompt then type

```
execfile('member.uid___A001_X1465_X1b48.scriptForPI.py')
```

Unlike for manually calibrated data, where scriptForPI.py re-runs the calibration by default, for pipeline calibrated data, scriptForPI.py will look for *casa_piperestorescript.py* and execute it. This will load the flagging and calibration tables included in the archival package and apply them to the raw MSs. The final calibrated MSs will be placed in the *calibrated* directory.

The *casa_piperestorescript.py* can also be run stand-alone, without the scriptForPI.py. See the pipeline documentation for more details.

5.4 Running casa_pipescript.py

In the case of pipeline calibrated data, there is an alternative way to generate the calibrated visibilities. It uses the script “*casa_pipescript.py*” that is provided in the *script* directory. This

contains the same tasks and parameters that were used when the ALMA QA2 processing calibrated the data. In principle, the user can modify this script to adjust the calibration, but it is recommended to try the delivered version first. The details of the usage of this script are described in the Pipeline documentation (see Section 6 below).

The script *casa_pipescript.py* can also be run via the scriptForPI.py. Simply move the *casa_piperestorescript.py* out of the *script* directory: scriptForPI will then use *casa_pipescript.py*. For some pipeline-calibrated data (e.g. TP data), the script *casa_piperestorescript.py* may not have been provided. In that case, the scriptForPI.py will run the *casa_pipescript.py* by default.

5.4.1 Considerations for Running Supplemental Manual Imaging Tasks

The user needs to be aware that if supplemental manual imaging was performed, like for any other imaging task, this will not be run by scriptForPI.py, which only deals with the calibration of the dataset. If the user wants to execute the supplemental imaging with different parameters, he/she will need to edit the scriptForImaging.py found inside the **manual_imaging.tgz** file located in the products directory. Since this is supplemental imaging, it is assumed that the imaging pipeline has been executed, therefore this scriptForImaging.py runs on the *_target.ms or *_targets.ms and *_targets_line.ms created by the imaging. To be able to run scriptForImaging.py in the case of supplemental imaging, the user will then need to restore the continuum subtraction and create the *_target*.ms. This restore can be done following the instructions at this link:
https://casaguides.nrao.edu/index.php?title=ALMA_Imaging_Pipeline_Reprocessing

See section Restore Pipeline Continuum Subtraction and Manually Make Image Products. In section 5.6, below, an alternative way employing the scriptForPI.py is provided. Once *_target*.ms files have been created, scriptForImaging.py can be run from within CASA. Since the data reducer may have used pipeline tasks to create the supplemental products, it is always better to load CASA with the --pipeline option, to execute scriptForImaging.py.

5.5 Saving disk space during and after the execution of the scriptForPI.py

The ALMA datasets are increasingly large. If the user would like to save disk space and only work with the calibrated data once these were regenerated with “scriptForPI.py”, they can either delete all calibration directories (cd calibrated; rm -rf *.calibration) or, in case of manually calibrated data, only delete some or all of the intermediate MSs contained in the calibration directories, using

```
cd calibrated; rm -rf *.calibration/*.ms; rm -rf *.calibration/*.ms.split
```

Since the middle of Cycle 1, the scriptForPI.py has also offered the “SPACESAVING” option to limit the disk space usage during and after the run its execution. In order to make use of this, the Python global variable SPACESAVING needs to be set before starting the script, e.g. using

```
cd script
casa -c "global SPACESAVING; SPACESAVING=N; execfile('scriptForPI.py')"
```

where N is an integer from 0 to 3 with the following meaning:

SPACESAVING	= 0	same as not set (all intermediate MSs are kept)
	= 1	do not keep intermediate MSs named *.ms.split
	= 2	do not keep intermediate MSs named *.ms and *.ms.split
	>= 3	do not keep intermediate MSs named *.ms, *.ms.split, and *.ms.split.cal (if possible)
	= -1	do not check disk space

With SPACESAVING=0, the required additional diskspace is up to 14 times as large as the delivered data (products and raw data), while with SPACESAVING=3 (maximum savings), it is up to 6 times as large. The script will estimate the required disk space and will not execute if there is not sufficient free space available. You can disable the disk-space checking by setting the value to -1.

5.6 Splitting out the calibrated data at the end of running the scriptForPI.py

For manually calibrated data, it is standard practice to conclude the calibration by copying the calibrated visibilities into a new MS at the end of calibration. This is done using the CASA tasks *mstransform* or *split* and is called "splitting out" the calibrated data. The final calibrated MSs are named *uid*.ms.split.cal.* They contain only the science spectral windows.

For pipeline calibrated and imaged data, the calibrated visibilities for the science target(s) are split to a new MS with **target.ms or *targets.ms* (the latter only from Cycle 9) to facilitate the continuation of the processing with the imaging section of the Science Pipeline. The original MS is called *uid*.ms* and contains all original spectral windows and both the calibrated and uncalibrated data. The calibrated data can be found in the MS table column "CORRECTED". Before Cycle 9, the split MS, called *uid*target.ms* contains two columns of data: the calibrated (continuum + line) data are in the DATA column and the continuum subtracted line visibilities are in the CORRECTED_DATA column ready for line imaging. From Cycle 9, two split MSs are provided: *uid*targets.ms* and **.targets_line.ms.* The split MS *uid*targets.ms* has the science target calibrated data (continuum + line) in the MS table column DATA which is automatically picked up by the CASA imaging tasks. The split MS named **_targets_line.ms* has the continuum-subtracted calibrated data of the science target in the DATA column. To create the split MS **.targets_line.ms* with the continuum-subtracted line visibilities in the DATA column, the task–uvcontsub has to be executed with the imaging pipeline. From Cycle 10, these MSs can contain a CORRECTED column with the self-calibrated visibilities if self-calibration was successful for the given target.

If you want to force the *scriptForPI.py* to perform the "splitting out", i.e. to create *uid*.ms.split.cal* for each EB, you can set the variable DOSPLIT to True before starting the script:

casa -c "global SPACESAVING; global DOSPLIT; SPACESAVING=N; DOSPLIT=True; execfile('scriptForPI.py')"

5.7 Generating the pipeline continuum subtraction products

Starting with data from Cycle 8, it is possible to make the scriptForPI.py also trigger the generation of the continuum subtracted MSs. This is controlled by the variable "DOCONTSUB". Just like with DOSPLIT above, you need to set the variable to True before running the scriptForPI.py using

execfile. From Cycle 10 onwards, DOCONTSUB will automatically be run if the PL used self-calibration.

5.8 Generating plots related to the renormalization task

Since Cycle 7, data suffering from incorrect amplitude normalization caused by bright astronomical lines detected in the autocorrelations of some target sources are adjusted when the correction fraction is more than 2%. More information on the subject can be found at the documentation of the ALMA Science Pipeline (see Section 6) and at the following knowledgebase articles:

https://help.almascience.org/kb/articles/what-are-the-amplitude-calibration-issues-caused-by-alma-s-normalization-strategy

https://help.almascience.org/kb/articles/what-errors-could-originate-from-the-correlator-spectral-normalization-and-tsys-calibration

In the pipeline a comprehensive development of this task is available only from Cycle 10. During Cycle 7, interferometric datasets were checked for the renormalization issue using an external script with CASA 6.1. The diagnostic plots (renorm scaling factors vs. frequency) can be generated by the user employing *scriptForFluxCalibration.py* executed by the scriptForPI.py: the diagnostic plots are found at the *calibrated/working/Plots* directory. In Cycle 8 and CASA 6.2, the renorm task was included within the pipeline. The diagnostic plots are accessible from the weblog at the hifa_renorm stage clicking the link to diagnostic plots provided in pdf format. Strong atmospheric lines and improper data segmentation may have caused a false triggering of the correction by the pipeline but adjusted manually by the DRs employing an external script. The required commands for a proper renormalization application are incorporated in the *casa_pipescript.py* and the diagnostic plots are reproducible on disk by the user: see *calibrated/working/RN_plots* directory. For full polarization datasets when performing renormalization in the *scriptForPolCalibration.py*, the scriptForPI.py shall be executed with the option SPACESAVING=1 to make sure the MSs are available for a successful renormalization application. During Cycle 9, strong ATM lines are automatically flagged by the pipeline, but false segmentation correction is still improved by the DR with a script using CASA 6.4. The diagnostic plots can be viewed in the weblog under hifa_renorm stage. Also in this case the *casa_pipescript.py* execution produces the diagnostic plots as previously described. For Cycle 10, no additional manual intervention for the renorm task by the DR is foreseen when employing the pipeline. Diagnostic spectra plots can be inspected in the weblog at the hifa_renorm stage.

5.9 Manually calibrated TP data

If your TP data were reduced manually, you will find scripts "*.*scriptForSDCalibration.py*" in the script folder. If the scriptForPI.py is also present, you can proceed by running it as for other data. For details on manual TP data calibration, please see the official CASA guide available at

https://casaguides.nrao.edu/index.php?title=M100_Band3

In older deliveries, the scriptForPI.py was not present. In that case, you need to download the raw data (ASDMs) from the ALMA archive, remove the '.asdm.sdm' extensions, and then run the scripts ('*.*scriptForSDcalibration.py*' from the scripts folder) in the same folder with the ASDM.

For the imaging, we recommend you use the provided script, as it contains certain values (e.g. Jy/K conversion factors) that must be set to a precise value. You could run the script step-by-step, as for the calibration script. Before doing that, you may need to update the paths to the calibrated data, in the msNames variable near the top of the script.

5.10 Pipeline calibrated TP data

Similar to the interferometer pipeline, there are two methods available for the user to restore pipeline calibrated TP data. The first method (A) requires rerunning the entire pipeline using scriptForPI.py and casa_pipescript.py, while the second method (B) involves using scriptForPI.py and casa_piperestorescript.py with some edits to restore the calibrated MS(s). For method (A), please refer to Section 5.3.

It is important to note that simply executing scriptForPI.py will NOT create the calibrated MS(s). Method (A) is recommended for users who are not familiar with the calibration pipeline, as it runs the entire pipeline and creates the calibrated MS(s). However, a disadvantage is that it takes longer than method (B), which restores the first part of the pipeline process using the existing calibration tables. Method (B) is especially useful for big datasets, such as having many EBs.

In the remaining part of this section, we will explain the steps for restoring calibrated TP MS(s) using method (B). To restore the calibrated MS(s) using method (B), the user employs the scriptForPI.py and casa_piperestorescript.py files located in the *script* directory. By executing scriptForPI.py (e.g., with execfile), casa_piperestorescript.py restores the calibrated MS(s) from the raw ASDMs by applying the necessary calibration tables. For TP data, casa_piperestorescript.py only restores the TP pipeline tasks from hsd_importdata to hsd_applycal. The calibrated MS(s) will be placed in a newly generated *calibrated* directory. To reproduce the same calibrated MS(s) created by QA2, the user needs to run the remaining pipeline tasks written in casa_pipescript.py (i.e., from hsd_atmcor to the 2nd hsd_blflag with pipelinemode='automatic'). It is important to note that the CASA version should be the same if an exactly same calibrated MS(s) is needed. In case the user wants to process the MS(s) in a different way than QA2 (e.g., skip hsd_atmcor), method (B) is useful.

6 Relevant documentation and help

ALMA Quality Assurance is described in chapter 11 of the ALMA Technical Handbook
https://almascience.org/documents-and-tools/cycle11/alma-technical-handbook

For more information on the usage of CASA, please refer to the CASA home page
http://casa.nrao.edu/

Detailed examples are given in the ALMA guides
http://casaguides.nrao.edu/index.php?title=ALMAguides

and the documentation of the ALMA Science Pipeline is available from
https://almascience.org/processing/science-pipeline

If you have any problems with your ALMA data, please contact the ALMA helpdesk at
https://help.almascience.org/

The Atacama Large Millimeter/submillimeter Array (ALMA), an international astronomy facility, is a partnership of the European Organisation 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 and Technology Council (NSTC) in Taiwan 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.