Data product naming requirements and recommendations

Version 4.8

Revision history

Version number	Note	Author	date
Original document	Presentation format	Lindsey Davis	
Ver. 2		Lindsey Davis	
Ver. 3		Lindsey Davis	
Ver. 4	Transferred to document format.	E. Muller	31st Aug 2011
Ver. 4.3	Minor tweaks following F2F	E. Muller	2013
Ver. 4.4	Minor tweaks for clarity & addition of new associations for image types.	E. Muller	2014
Ver. 4.5-4.7	Updated imagetype (add full/lowres), add comments from Nagai, Villard & Petry, Hubbar, Hunter, Indebetouw etc. (see SCIREQ-110)	E. Muller	Apr 2015
Ver 4.8	Additions following comments from Pipeline (SD & interf) sec 9.4 - Additions to POL types remove sdcal datatype	E. Muller, K Nakanishi, R. Miura, etc.	Jan 2016

Contents

1. Scope

This document contains the recommendations and requirements for ALMA data products of the Inter-ARC Science Archive Working Group. The recommendations made here are formulated by the ALMA Science Archive Working Group (ASAWG) after consultation throughout the members' respective ARCs, with the considered and assumed preferences of the ALMA user-base in mind.

2. Motivation

The recommendations made here are geared towards simplicity, consistency and practicality for ALMA-users to easily process, analyse and book-keep their data; providing a naming scheme that describes file contents at-a-glance; preserves data created from previous reduction/processing attempts and avoids complications associated with meta-characters. The overarching consideration is the long term archiving and simple retrieval of ALMA data products for future researchers. The convention is to to be applied to products produced by the data processing pipeline: data reduction and data products, including Measurement sets, flagging tables, calibration tables, images, CASA logs, Web logs i.e. (Archive UID(s) <-> Disk path / file name(s) translation are to be supported).

We recommend file and directory names be absolutely unique, reproducible (i.e. in subsequent processing iterations of the same data), descriptive, and practical to enable understanding, browsing, pattern matching, sorting. Effects of naming complications should be minimized for the user, E.g. Iterations, parallelization processing should be transparently and intuitively labelled and identified. The file packaging and contents naming will be complete and deterministic: i.e, any two identical processing runs made on the same dataset will yield identical contents, including the naming structure.

3. General nomenclature guidelines

General data product naming recommendations are made as follows. Where necessary, details on each item can be found in the remaining document sections.

- 3.1 All names must be as compact as possible, consistent with clarity, and avoid meta characters [\b, /, \, [], =, ~, *, :] that interfere with pattern matching in commonly-used computer operating systems: Linux, DOS (see section 4.).
- 3.2 Lower case characters must be used in all file, source and calibrator target names.
- 3.3 Data structure must be hierarchical, and emulate structure of telescope control command grouping (see sections 5, 6 7, 8 & 9).
- 3.4 Data products must follow the association precedence order defined for each type of data product (see Sec. 7.).
- 3.5 Names must indicate content format e.g. ASDM,table etc. (see Sec. 9.1- 9.7)
- 3.6 Names of calibration data must describe data context. e.g. phase, gain, amp. and indicate data function (see Sec 9.4)
- 3.7 Names must indicate data format. E.g. fits, png, html, txt, tar, tgz … (See Sec. 9.5.)
- 3.8 Web log names should reuse the science data product conventions where consistent with other goals (see Sec. 9.6.).
- 3.9 Processing software logs should be preserved in delivered data product (see sec 9.7)
- 3.10 Use of internal tags should be minimized and restricted to that easily understood by the user, e.g. stage number, iteration counter, sequence number. Acceptable instances of internal tags:
  - 3.10.1 iteration counters where appropriate, e.g. the single dish iteration loop.
  - 3.10.2 sequence numbers: e.g. monitoring programs.
- 3.11 Internal tags must be removed from final data product names.
- 3.12 Distinguish temporary data products, e.g. loops over bandpass methods (to determine best bandpass solution), from permanent ones by using sequence counters and names consistent with file naming schemes outlined in sections 8 and 9.
- 3.13 Temporary files resulting from interim processing stages shall not contribute to the final delivered data product.
- 3.14 Make parallellization as transparent to the user as possible. Use reference ASDM name, not sub ASDM names.

4. Name usage and abbreviation guidelines

- 4.1 Target source names should be labelled consistently with nomenclature used by PI, despite any nondescript meaning and also in the case where the target name is clearly identifiable as a

misrepresentation of a name easily resolvable by SIMBAD. e.g. “source_1” is a valid target name and should be preserved. Equivalently, while “SagAstar” might be resolved into SagA_str, the nomenclature preferred by PI should be used.

4.2 Else, source names should be formatted to be resolvable by SIMBAD, including all calibrators

4.3 The source names should only contain the following characters: a-z, A-Z, 0-9, -,+, _ or a . (full stop). characters: e.g.[\b, /, \, [], =, ~, *, : space] or other text that may be interpreted by e.g. linux or other common operating systems must be modified according the table on the right.

[,]	**_ (underscore)**
/,	**_ (underscore)**
\b	**b**
=	**eq**
~	**tild**
*****	**str**
:	**_ (underscore)**
space	**_ (underscore)**

5. Observing unit set structures

5.1 The SB is the smallest unit of data that must be taken together.

5.2 The OUS is the smallest unit of data that should be calibrated together.

5.3 Every observing project contains 3 observing unit set levels labeled science goal, group, and member. This structure is illustrated in Figure 1.

5.4 The science goal observing unit set is a container for 1 or more group observing unit sets. Pipeline processing is never enabled at this level.

5.5 Group observing unit sets are containers for 1 or more member observing unit sets. Pipeline processing may or may not be enabled at this level. If enabled intermediate results from the member observing units sets are combined to form the final data products.

5.6 Member observing units sets are containers for scheduling blocks. In the majority of cases the number of scheduling blocks will be 1. Pipeline processing is always enabled at this level.

5.7 Pipeline processing is triggered at the individual member observing unit set level and at the individual group observing unit set level. In both cases processing proceeds from the member to group observing unit set level.

6. Directory structures:

6.1 The pipeline must establish a working directory structure for processing the data. The archive must establish a working directory structure for data delivery to the users. These directory structures should be as similar as possible to each other and reflect the project structure.

6.2 Figure 2 illustrates the proposed pipeline working directory tree. The top 3 layers parallel the top 3 layers of the project structure shown in Figure 1.

6.3 Below each group and member observing set directory is a data, working, and products directory. Data will be imported from the archive to the data directory, filled and processed in the working directory, and exported to final data product or uploaded to the archive in the products directory.

6.4 This structure supports both multiple automated pipeline runs and multiple interactive runs.

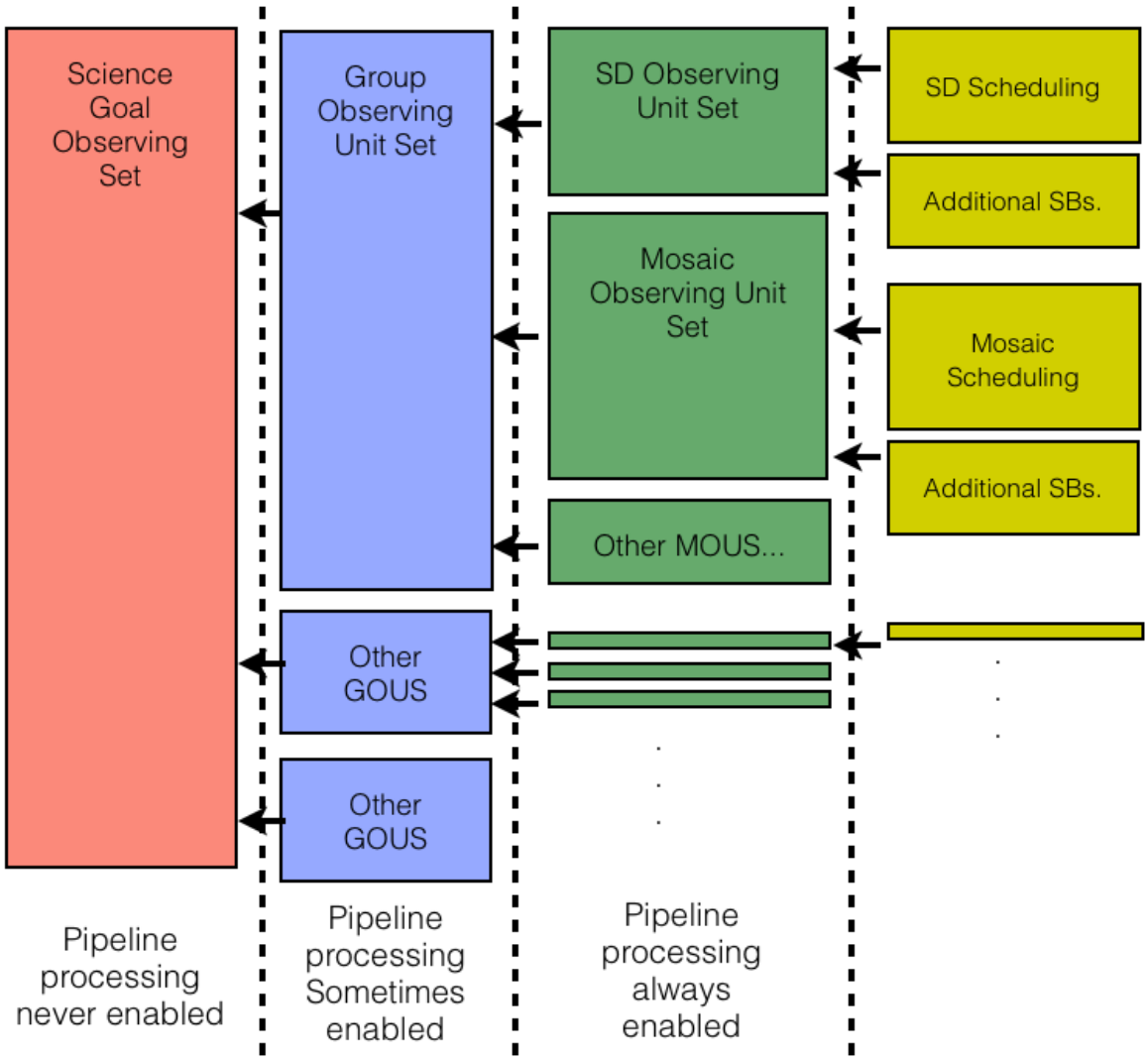

Figure 1.
Diagram of basic observations control hierarchy. The simplest element is the Scheduling block, a number of which may be grouped into MOUS and then into GOUS, to acquire data to address a PI-defined Science Goal. Heuristic processing scripts may be applied to data collected and grouped at the MOUS or GOUS level.

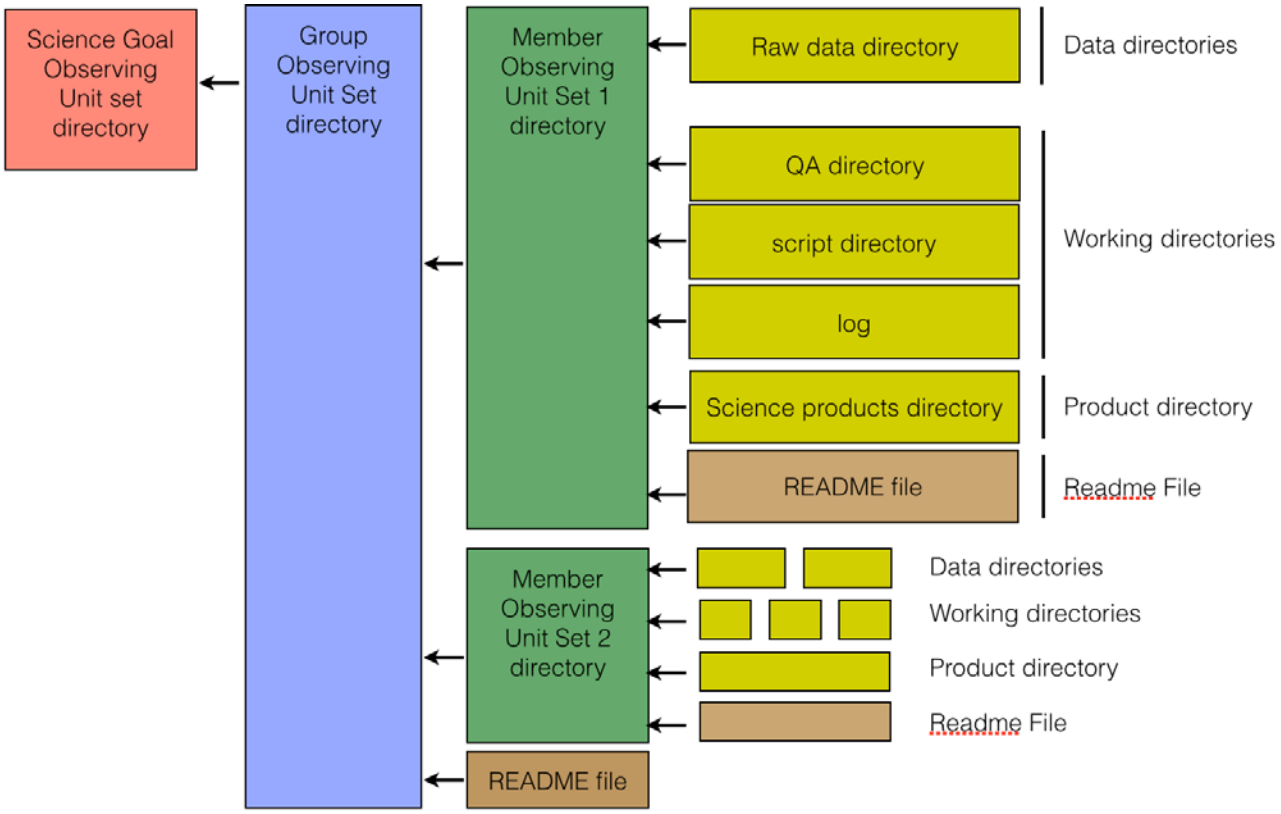

Figure 2.
Basic data product structure for the simplest data package comprising processing of data from only two scheduling block elements (comprising e.g. two targets, or different correlator setups).

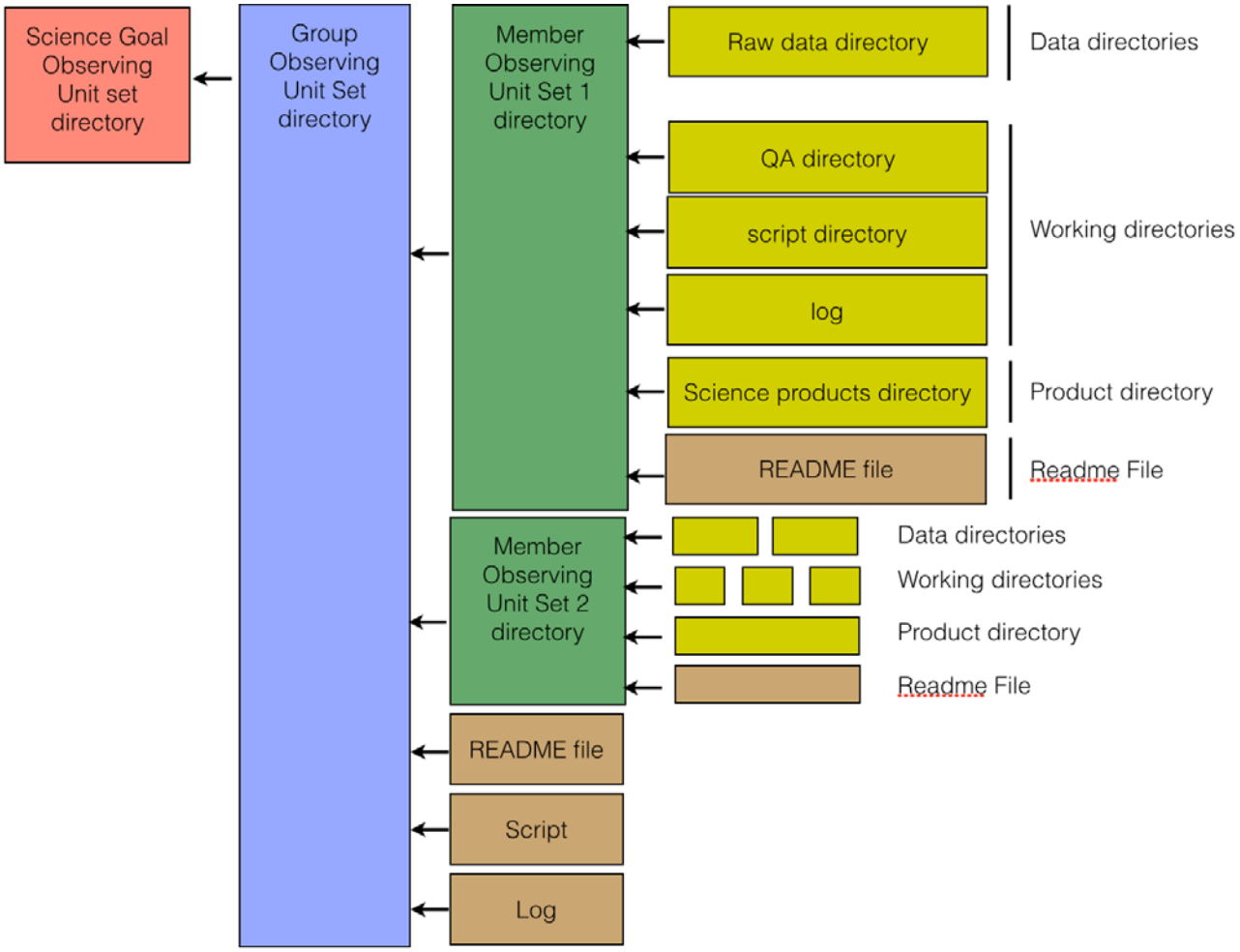

Figure 3.
Data product structure in the case where SBS are to be combined at the GOUS level - e.g. an SB of single-dish + an SB of Interferometer datasets. In this case, an additional “products” directory containing the processed combined dataset and README is included at the GOUS level. The README file at the OUS level details the file listing of scheduling blocks as for figure 2, The logs and scripts for the combination is kept at the GOUS level.

MOUS-level:
<ROOTDIR>/science_goal.<SGUID>/group.<GOUS_UID>/member.<MOUSID>/<raw | calibration | product | qa | log | script | member.<MOUSID>.README.txt

GOUS-level- (i.e. for data products where MOUS are combined):
<ROOTDIR>/sciencegoal.<SGUID>/group.<GOUS_ID>/ <raw | calibration | product | qa | log | script | README.txt>

7. Data product path naming syntax

7.1 Path naming convention syntax should have the general format of:

7.2 ROOTDIR name must be unique for all pipeline runs, but subdirectories names don't change between runs.

7.3 Subdirectory names reflect level of structure (Group, member, etc; see Sec. 9)

<association1>.<association2>…<associationN>.<type>.<format>

7.4 File name convention syntax:
Associations are dependent on the data product and specify concepts such as parent Execution Block, source, spectral setup, etc. At least one worldwide-unique association is required for all final data products.

Type specifies the type of data product, e.g. a gain calibration or a cleaned image. Type or format is a required field for all final data products.

Format specifies the storage format of the data, e.g. a text file, a FITS image, a png graphic, a tar file, a CASA table, a CASA image, etc. Type or format is a required field for all data products. Type and format should be retained, even in redundant cases (e.g. ASDM, see Sec. 11).

8. Observing unit set subdirectory names (below MOUS/GOUS level)

8.1 The observing unit set directory names must be unique at each level in the directory structure. It

<path>/group.<sanitized group OUS uid>/<remainder>
<path>/group.<sanitized group OUS uid>/member.<sanitized MOUS uid>/ <remainder>

is the responsibility of the pipeline to create the OUS subdirectories

8.2 Value to enable easily recognized names & data structure by PIs,

9. Data product naming conventions.

9.1 Execution Block (ASDM-format data) naming convention

9.1.1. Creating execution block file names is the responsibility of the Pipeline / Archive ASDM retrieval code.

9.1.2. Disk name and the heuristics code should accept as input ASDM names which do not conform to this standard, as currently (cycle 0) the ASDM export task does not add a type or format identifier to the execution block.

Associations	ASDM UID
Type	“asdm” (ALMA Science data model; redundant with format)
format	“sdm” (Science data model: redundant with type)
syntax	<ASDM UID>.<type>.<format>
E.g.	uid___X02_X3d737_X1.asdm.sdm

9.2 Flagging table naming convention

9.2.1. Creating flagging table names is the responsibility of the Pipeline heuristics code.

9.2.2. The final data product must be a single table of processing flags per execution block that can be applied directly to the raw visibilities.

9.2.3. Internal tags must be prefixed or appended to the flag table name for easy removal.

9.2.4. Along with calibration tables, flagging tables will be included in a single TAR or similarly-compressed archive format file and compressed (or “zipped”).

Associations	ASDM UID
Type	“flags”
Format	tbl (for “table”)
syntax	<ASDM UID>.<type>.<format>
E.g.	uid___X02_X3d737_X1.flags.tbl

9.3 Calibration tables naming convention

9.3.1. Creating calibration table names is the responsibility of the Pipeline heuristics code.
9.3.2. The final data product should include a set of calibration tables which can be applied to the execution blocks to produce calibrated visibilities.
9.3.3. Fit type should describe the degree of polynomial (poly_n; where 'n' is the degree of the polynomial), or the function type used to fit.
9.3.4. Internal tags must be prefixed or appended to the calibration table name
9.3.5. Spectral information is given as a band number (B#) followed by a list of underscore-separated spectral window numbers: 'B3_1_3_4'
9.3.6. Polarization calibration tables; cross-hand delay (XY delay), cross-hand phase (XY-phase) and instrumental polarization (D-terms) are identified thus: cdcal, cpcal, dcal
9.3.7. Along with flagging tables, calibration tables will be included in a single TAR or similarly-compressed archive format file and compressed (or "zipped").

Associations		
	freq_info	<B3_1_3_4>
	fit_type	< "poly_n" | "spline" | "tseries" | … >

Type	< "bcal" | "gcal" | "gacal" | "gpcal" | "tsyscal" | "wvrcal" | "cdal" | "cpcal" | "dcal" | "spblcal" | "skycal" >
format	"tbl"
syntax	<ASDM UID>.<freq_info>.<fit_type>.<type>.<format>
E.g.	uid___X02_X3d737_X1.90_8.poly_3.bcal.tbl

9.4. Image naming convention

9.4.1 Creating image names is the responsibility of the Pipeline heuristics code.
9.4.2 The final data products must include image cubes in FITS format
9.4.3 Images produced for short-term monitoring projects are explicitly counted in the image name.
9.4.4 The source specification will include a unique combination of source name, observations intent and iteration counter (relevant only for short-term monitoring observations, i.e. counter is always equal to zero for non-monitoring projects and calibration intent data).
9.4.5 Source names must be derived from the execution block/AOT (i.e. the ALMA name).
9.4.6 Long intent names must be shortened: e.g. 'CALIBRATE_BANDPASS' to 'bp'. Multiple intents will be concatenated with '_' as delimiter.
9.4.7 Dimensionality is exclusive of stokes (e.g. Position-Position-Velocity cube is 'cube', or Position-Position image is 'image').
9.4.8 Spectral information is given as a concatenation of spectral window number, and frequency of the spectral window centre (in GHz, with 6 significant figures) :<spw#>_<freqGHz> . Continuum maps compiled using all spws, use spw#=0, and a central frequency that is the geometric mean of all spectral windows.
9.4.9 Cubes will be produced at the native (full) spectral resolution. If the PI-requested resolution differs from native by ≥4x, then another, lower-resolution cube is generated with the spectral resolution specified by the PI.
9.4.10 Polarisation will be stokes parameters, or polarisation flux 'P', or polarisation angle 'A', these are 'flux' datatypes.
9.4.11 Image types of 'mask' and 'flux' will only appear with datatype 'clean'

9.4.12 Internal tags should be prefixed / appended to the image table name

9.4.13 "arrays" reflects the arrays contributing to the group level, and may be concatenated (i.e. one of "12m", "7m", "TP". or "12m7mTP", "12m7m"). NB. will only be one value at Member level.

9.4.14 *How do we deal with primary beam corrected images? KN suggestion: adding '_pbcor' to datatype association for a primary beam corrected image, e.g. "contsub_spec_pbcor"*

(Table on following page)

	OUS rank	< "member" | "group" >
	OUS ID	< MOUS/GOUS ID >
	Source specification	< sourcename>_< intent >_< monitoring_counter > Intents: <"chksrc" | "bp" | "gain" | "amp" | "phase" | "polleak" | "polang" | "sci"> (checksource, bandpass, gain, amp, phase, pol leak & angle calibrators; science target). N.B. intents may be compound, e.g. "bp_phase". See 9.4.6
Associations	freq_info	<spw#>_<spw_centre_freq(GHz)>
	pol	Polarization: < "I" | "Q" | "U" | "V" | "P" | "A"> or combination
	obspatt	Observing pattern: < "sd_raster" | "sd_fm_lissajou" | "sd_fm_doublecircle" | "sf" | "mos" > Single dish, or else single field or mosaiced interferometer pointing.
	arrays	one, or concatenation of strings indicating the array used: <"12m" | "7m" | "TP">
	dim	Dimensionality: < "cube" | "image" >
	resn	Native or PI-requested resolution <"full" | "lowres">, must be "lowres" for 2d images
	imagetype	< "dirty" | "clean" | "resid" | "model" | "sd" | "mask" | "flux" > dirty, clean, residual & model images (for interferometric obs); single dish & single dish calibrator
	datatype	< "flux" | "mask" | "cont" | "spec_contadd" | "spec_contsub" | "psf" | "mon*N*" > Flux, mask, continuum, continuum+spectral, cont-subracted spectral, PSF (beam) and *N*th-order moment image, where *N*=0,1,2.

Format	< "fits" | "cim" > (FITS format | CASA image format)
Syntax	<member | group>.<m/gousID>.<sourcename>_<intent>_<monitoring_counter>. <freq_info>.<pol>.<obspatt>.<arrays>.<dim>.<resn>.<imagetype>.<datatype>. <format>
E.g.	member.uid_A000_X00_Xxx1.3C454_sci_0.0_235.123456.I.sf.12m7m.image2.lowres.clean.cont.fits

9.5 Web log file name convention

9.5.1. Where appropriate file names and links should follow science data product name conventions, e.g. use of execution block UID, source specification.

9.5.2. Syntax of compressed html tree tar file: <html.tar.gz> e.g. html.tar.gz

Associations	OUS rank	< “member” | “group” >
	OUS ID	< MOUS/GOUS ID >
	origin	< “weblog”>

Type	Follow web and graphics standards, e.g. “html”, “png” where possible
Format	“tar.gz”
Syntax	<member|group>.<origin>.<type>.<format>
E.g.	uid___A001_X122_X133.weblog.html.tar.gz

9.6 CASA log, PPR and QA report file naming convention

9.6.1. Pipeline processing requests and pipeline processing results captured respectively as “ppreq” and “ppres”.

9.6.2. QA reports at QA2 and QA0 are respectively of origin “qa2” and “qa0”

9.6.3. CASA Logs are always origin “casapy_log” and type “log”

9.6.4. Origin “ppreq” and “ppres” are always type “ppr”

9.6.5. Origin “qa2” and “qa0” are always type “rept”.

9.6.6. CASA command logs are always origin “casa_commands” and type “log”

Associations	OUS rank	< “member” | “group” >
	OUS ID	< MOUS/GOUS ID >
	origin	< “casapy_log” | “casa_commands” | “pipeline_manifest” | “qa2” | “qa0” | “ppreq” | “ppres”>

Type	< “log” | “rept” | “ppr” | “xml” >
Format	“txt”
Syntax	<member | group>.<m/gousID>.<origin>.<type>.<format>
E.g.	member.uid_A000_X00_Xxx1.casapy_log.log.txt

9.7 Pipeline script file naming convention

9.7.1 CASA pipeline scripts (“casa_pipelinerestorescript” and “casa_pipescript”) are always type “py”

Associations	OUS rank	< “member” | “group” >
	OUS ID	< MOUS/GOUS ID >
	origin	

Type	< “casa_pipelinerestorescript” | “casa_pipescript” >
Format	“py”
Syntax	<member | group>.<m/gousID>.<type>.<format>
E.g.	member.uid_A000_X00_Xxx1.casapy_pipeline_restorescript.py