The ALMA QA2 Imaging Script Generator almaqa2isg.py (ISG) in order to speed up and standardise the generation of scripts used for manual imaging. It is maintained by D. Petry <dpetry@eso.org> . On this page you find the documentation of the public version of it (which may be a version or two behind the latest version in use at the ALMA observatory).

The ISG takes as input only a few parameters like the names of the MSs to be imaged and creates a "scriptForImaging.py" tailored to the needs of the particular case. The result is a nearly complete script draft which the analyst then only needs to complete and adapt in a few places. At the end of the run, the ISG gives in the terminal output a list of the most important adaptions which the analyst still should apply.

The resulting scriptForImaging.py employs the same stepping mechanism as the manual calibration scripts produced by the Calibration Script Generator. It also follows all standards for the image naming.

The ISG is fast and, in simple cases, does not take more than a minute to generate the imaging script draft. It can be used for all observing modes. There is support for the alignment of shifted SPWs from different EBs using mstransform, for continuum subtraction, for the imaging of calibrators, and, starting with CASA 5.4, for the imaging of ephemeris object science targets.

Support for full polarisation was added in version 1.18.

Support for self-cal, and possibly for TP imaging is planned for the future. The ALMA Calibration Script Generator (public version) already contains simple support for TP imaging.

The ISG internally uses the analysisUtils and needs CASA 5.1.1 or later. CASA 6 is recommended.

Access

The ISG is contained in a single Python module **almaqa2isg** which is part of the analysis_scripts.tar package which also contains the "analysisUtils" which you will also need since the ISG uses a few of their functions.
See the Zenodo analysisUtils page (which provides a **citeable DOI**: 10.5281/zenodo.7502160) and the **Analysis Utilities CASA Guide** for details on how to install the package.

The first ISG alpha version was 1.7 . Described here is version 2.2 from 2022/12/19 (you can query the version using the method isg.version()).

Usage

With your sys.path set up to import the analysisUtils, type

```
import almaqa2isg as isg
help isg.generateImagingScript
```

to obtain the following help:

```
Help on function generateImagingScript in module almaqa2isg:

generateImagingScript(vis='', draft_threshold='', reqchanwidth=1, makecubes=True, docontsub=None, chanwidthtol=0.
05, restfreqs={},
                      additionalimg=['CALIBRATE_POLARIZATION', 'OBSERVE_CHECK_SOURCE'], spwmap=[],
perchanweightdensity=False)


    The ALMA QA2 Imaging Script Generator

    vis - the MS(s) to image, can be a list.
          If wildcards are used, a list is created automatically.

    draft_threshold - the cleaning threshold to use as a initial value in the cleaning (as a string e.g. "5 mJy")

    reqchanwidth - the requested channel width (for cubes) as a string (e.g. "200MHz", "20km/s")
                   or as an integer which indicates units of original channels;
                   if == None, the draft_threshold is used for the agg. bw.
                   default: 1

    makecubes - Boolean to indicate whether cubes should be made (set to False for continuum only)
                default: True

    docontsub - Boolean to indicate whether continuum subtraction should be done
                (only relevant if makecubes==True)
                default: True for standard polarisation, False for full polarisation data sets

    chanwidthtol - the tolerance in units of channel widths to apply when deciding whether the
```

```
                grids of corresponding SPWs from two different MSs are aligned
                default: 0.05 (i.e. 5%)

    restfreqs - the restfrequencies to be used for the restfreq parameter in the cube cleaning
                in the format of a dictionary, e.g. {25: '230GHz', 27: '231GHz', ... } with
                one entry for each science SPW (optional).
                default: {} (empty dictionary; the restfreqs, if needed, will be extracted from
                            the SOURCE table or, if not available there, the central freqs will be used)

    additionalimg - optional list of additional intents to be imaged.
                 For each field found with the given intent (give name omitting the "#ON_SOURCE"),
                 code for making an aggregate bandwidth image is going to be added.
                 Possible values: 'CALIBRATE_PHASE', 'CALIBRATE_BANDPASS', 'CALIBRATE_FLUX',
                      'OBSERVE_CHECK_SOURCE', 'CALIBRATE_POLARIZATION'
                 default: ['CALIBRATE_POLARIZATION', 'OBSERVE_CHECK_SOURCE'] (image check source(s) and/or polcal)
                 example: ['CALIBRATE_BANDPASS', 'CALIBRATE_PHASE']

    spwmap     - SPW IDs to be used in the final image names.
                 If a non-empty list is specified, it must contain at least as many
                 elements as there are science SPWs in the input MS. The given SPW IDs are then
                 used to replace in the image names the ones used in the MS,
                 e.g. if there are science spws [0,1,2,3] in the MS,
                 then setting spwmap=[19,21,23,25] will result in ID 19
                 being used instead of 0 in the image names, 21 instead of 1 etc.
                 default: [] (empty list) - use the SPW IDs as they are in the MS.

    perchanweightdensity - the setting of the tclean parameter perchanweightdensity
                 for "briggs" weighting. Starting with CASA 6.2, "briggs" weighting will
                 only be used for mfs imaging. For cubes, the weighting option "briggsbwtaper"
                 will be used and perchanweightdensity will be forced to True.
                 default: False

    Example:
      import almaqa2isg as isg
      isg.generateImagingScript('uid*.ms.split.cal', draft_threshold='0.1mJy', reqchanwidth='20km/s', spwmap=
[17,19,21,23])

    Discussion:
      You can run the Imaging Script Generator (ISG) like so:

         import almaqa2isg as isg
         cd <the directory where the MS(s) are which you want to image, e.g. "calibrated">
         isg.generateImagingScript(vis='uid___A00*.ms.split.cal', draft_threshold='0.1mJy', reqchanwidth='20km/s')

      The vis parameter can be a single string or a list. If it is a list, each element must be the name of an
      existing MS. If it is a single string, it can either be a single name of an existing MS or an expression
      using wildcards "*" or "?" to specify a group of MSs. This will internally be converted to a list of MSs.

      If makecubes is true (default), code for cleaning cubes for each science SPW is created.

      If docontsub is true, code for continuum subtraction (separate for each science field) is created.
      This parameter is only active is makecubes is true.

      draft_threshold specifies the threshold which is entered into the cleaning commands. It is related to
      the required rms from the MOUS proposal. It is used as the cleaning threshold in cube
      cleaning if makecubes is true (then the threshold for the agg. bw. image is left for editing by the
analyst).
      If makecubes==False, then draft_threshold is used as the cleaning threshold for the agg. bw. image.

      reqchanwidth is the bandwidth on which draft_threshold is defined. It is used as the channel width for the
cubes if
      makecubes is true.

      Like for the calibration script generator, the ISG is not meant to produce a script which one can run
      blindly. We are not trying to reinvent the imaging pipeline. So, if different channel widths are required
      for different SPWs, the analyst needs to edit the channel width (and the corresponding thresholds) for some
      of the cubes. A general solution to permit different thresholds and widths for different SPWs was deemed
too
      complex for the moment.

      NOTE that if there is more than one input MS and docontsub is True, then a concat step will be inserted
such
      that the uvcontsub commands can operate on the concatenated MS. If the SPWs are not yet reindexed to have
      IDs starting at 0, uvcontsub will do this reindexing. The ISG will take this into account and use reindexed
      SPW IDs when making the cubes but leave the old SPW ids in the image names (this is what archive needs).

      Furthermore, if constinuum subtraction is requested, the ISG will check if the corresponding SPWs of the
MSs
      are shifted w.r.t. each other and if so, find the largest common grid for each SPW and generate mstransform
code
      to transform all science SPWs of all the MSs into these grids before concatenation.

      At the end of the isg run, you obtain a file "scriptForImaging.py". On the terminal (in CASA) you will also
      get messages which explain what you need to do to complete this script before you can run it. These
```

```
messages
      look, e.g., like this:

        Script generation completed. Please find scriptForImaging.py in the current directory.

        NOTE that you still need to edit the script:
          - You need to edit the array "therestfreqs" to set the rest frequencies for the cubes of each SPW.
          - You need to adjust the threshold for the aggregate bandwidth/continuum image(s).
          - You need to edit the fitspw parameter in each uvcontsub command.

      Iterative use for continuum identification
      ------------------------------------------

      If cubes are to be created and a decision needs to be taken about whether contsub is necessary and how it
      should be done, one generally can go two routes:

      a) use plotms to generate amp vs. channel plots to identify lines and line-free regions

      b) create image cubes for each spw once to see if they contain detectable spectral lines and identify
         continuum channels.

      When going route (b), it is recommended that the user runs the ISG twice, first with docontsub=False and
then
      with docontsub=True: With the first version, one can create some shallowly cleaned cubes to check for lines
      and then move the first script version and the created images to a different directory and start again with
      docontsub=True.

      In order to make sure that a new run of the ISG does not destroy your previous version of the script, the
      ISG prevents you from overwriting a pre-existing scriptForImaging.py

      SPW Numbering
      -------------

      If you have used split or mstransform *with* reindexing to split out certain SPWs to obtain the input MS(s)
to the ISG,
      i.e. if your input MS(s) do *not* use the same SPW numbering as the original ASDM, you need to use the
parameter
      "spwmap" to get the archive-compliant numbering in the image names.
      The spwmap parameter should contain an ordered list of the *original* SPW IDs of *all* the science SPWs
      in the input MS(s). The input MSs, if there is more than one, need to all use the same numbering.
      A typical example is [19,21,23,25]. You need to look at the original raw MS after importasdm to obtain this
information.

      Full Polarization support
      -------------------------

      The ISG will detect whether an input MS contains full polarisation data. In this case it will use
stokes='IQUV' and
      dconvolver='clarkstokes'. Furthermore, if the intent CALIBRATE_POLARISATION is present, it will by default
make
      the appropriate calibrator image and add operations to calculate polcal parameters.
```

Feedback, bug reports, feature requests

Feedback, bug reports, feature requests should be made by email to dpetry@eso.org .