# Self-Calibration documentation page

This page is concecrated to the self-calibration tool for HBA observation with LOFAR. Here is explained what is 'the self-calibration' strategy, how it works, which tools are used and what are their parameters.

## Self-Calibration Global Strategy/Context/Ideas

The self-calibration process goals to improve the final quality of an image by an iterative process (N cycles). In our case (LOFAR data), the amplitude calibration by using a calibrator (target whose the flux is well known) is done automatically by the Radio Observatory (RO) in the imaging pipeline (see target and calibrator pipeline). By consequence, the self-calibration with LOFAR data is phase calibration ONLY.
At each cycle, self-calibration process improves the phase calibration of the data. To improve phase's data, we need at each cycle a more accurate sky model than at the previous cycle. To do that, the strategy is to image data at low resolution and extract a sky model, then calibrate data with this extracted skymodel. At each cycle, the image resolution increase with the goal to extract a more complete sky model for calibrating data with this new sky model at the next cycle.
This process is repeated until the image resolution obtained is the best resolution possible with the data (α longuest baseline)

## Release Notes (Installed version only on CEP1 (DOWN after the 24/11/2014) , CEP2, Flits, NOT yet on CEP3)

- **04/2014: Selfcal_v.2 (STANDALONE):** Major improvements, parameters becomes flexible as options. CEP and Flits version are the same (merged version). There is no more memory limitation (it does not degrade the resolution as before, but it is user responsability to use it properly, but of course warnings advertise in case of huge possible memory consumption or time computing. See the formula at the end of the help when you type selfcal.py -h)
  - **Parameters changed**
  - VLSSSuse becomes VLSSuse
  - outerFOVclean becomes outerfovclean
  - **Internal Changes**
  - Only obsDir, outputDir have to provided. Others parameters have a default value if not provided.
  - CORRECTED_DATA is not transfered to DATA, Original data is used for calibration at each cycle
  - For outerfovclean option, it is possible to provide the annulusRadius option to select the outer FOV region to independant direction peel. default is 1 degree (if outerfovclean=yes and annulusRadius not provided)
  - psf_vary_do is enable for source extraction with pybdsm (using psf_vary_do='True', psf_stype_only='False', psf_snrcut=5, psf_snrcutstack=5, psf_snrtop=0.3)
  - RMS_BOX_Bright = 40,10 for bright sources only and adaptive_thresh=30 (adaptative_rms_box module in pybdsm)
  - masking is enable by default (i.e, at each cycle: calibration, imaging, src extraction, mask

generation, imaging with mask, final src extraction instead: calibration, imaging, src extraction ⇒ longer time computing)
  - Image number of pixel is composite (multiple of 2^(m).3^(n).5^(p). Make imaging faster.
  - Possibility to provide parameters through a parset file (see documentation or type selfcal.py -h to access the help)
  - to access to the help type: **selfcal.py -h**
  - **Resolution selection: 4 cases**
  - nothing provided: go to the best resolution available. Starting at 15 times the best resolution, and iterate with linear steps.
  - startResolution and endResolution provided: linear steps from startResolution to endResolution (depends the nbCycle parameter)
  - startingFactor and endResolution provided: same as above, just the startResolution = startingFactor x endResolution
  - resolutionVector provided: must be on this shape: resolutionVector=x1,x2,x3,...,xn . each cyle has its resolution (in arcsec) frozen by the user.

 Asap a new version with peeling option will be provided.

- **11/2014: MergeSB_v.1 (STANDALONE):** New option nofChannel2Merge (if not provided, default is 1 channel per subband) is implemented.

 Another one is checkNames but not activated yet. It will allow to use mergeSB.py with files using a different nomenclature as intermeadiate or final data nomenclature (but the user will have to provide some informations which cannot be extracted from nomenclature).

- **09/2014: Observatory Selfcal_v0 (RADIO OBSERVATORY):** Final successfull tests (using regression tests. Use same parameters as the standalone **Selfcal_v.1** and the standalone **MergeSB_v.0.1** except that at each iteration CORRECTED_DATA is not transfered to DATA, Original data is used for calibration at each cycle.

 at this day: 11/2014; Problems to implement it as a pipeline. Code is ready, but implementation must be done at least for LOFAR cycle 4: 04/2014.

- **05/2014: MergeSB_v.0.1 (STANDALONE):** Replace Madflagger by AOFlagger have a better flag (avoid to flag all datas)

- **04/2014: Selfcal_v.1 (STANDALONE):** New parameters implemented (must be provided): outerFOVclean and VLSSSuse. Allow an independant direction peeling (at 90 arcsec resolution) using an outer fov annulus radius. This radius is estimated automaticaly estimated from average elevation observation. Possibility to use the extracted sky model at 90 arcsec resolution (from the preprocessing using outerFOV option) to start selfcal. Number of pixel per beam changed from 2.5 to 4. Use in pybdsm extraction: ini_method=curvature, adaptative_rms_box for better extraction. Replace in BBS parset `Step.solve.Solve.Mode = PHASE` by `Step.solve.Solve.Mode = COMPLEX`, it avoids to divide by the SNR at each iteration.

- **11/2013: Selfcal_v.0 (STANDALONE):** First version avaible. Finalize extraction parameters (used in pybdsm: filename, detection image, thresh_isl=6, thresh_pix=8, RMS_BOX=[80,10],

atrous_do=True). UVmin estimated (0 klambda if dec >35 degree; 0.1 klambda else). CORRECTED_DATA column is transfering to next iteration as DATA column. FOV is frozed to 5 degree. Parameters must be provided (obsDir, outputDir, nbCycle). Two different versions for Flits and CEP1-2. Memory limitation factor: decrease automaticaly the final resolution if data size is too voluminous. Final resolution limited to 10 arcsec on CEP, no limitation on Flits.

- **11/2013: MergeSB_v.0 (STANDALONE):** Merge subbands MSs for each time chunk in one MS. Merge automaticaly to one channel per subband. Accept only intermediate and final data format (see use below). Parameters must be provided (obsDir, outputDir,column2Merge)

# Self-Calibration in the Radio Observatory imaging pipeline

The self-calibration is currently being implemented in the pipeline of the radio observatory. For testing purposes, a stand-alone tool is available. .....in processing

# Self-Calibration Standalone Tool

The self-calibration standalone tool is installed on CEP1, CEP2 and on Flits (the ASTRON Astronomy group server).

To use it on CEP, you can run selfcal by running

```
/opt/cep/selfcal/selfcal.py
```

You can load an initialization file to avoid typing the path: `source /home/vilchez/selfcal/selfcal_init.sh`

On Flits, just type

```
selfcal.py
```

Launching 'selfcal.py' without arguments will print short usage info:

```
MISSING Parameters
Usage: selfcal.py --obsDir= --outputDir= --nbCycle=
```

#### selfcal.py parameters

- `obsDir`: a directory containing ONLY the Measurement Sets (MS) for processing. Usually, this directory contains all time chunks for an observation. The time chunks could be merged in frequency by containing several subbands. If time chunks are merged in frequency, be careful that each time chunk contains the same subbands. The path of `obsDir` must be an absolute path (no use of ~/mypath/).
- `outputDir`: the output directory, where the self-calibration process will write results. If it does not exist, it will be created. Just be sure that you have rights to write at this place. The path of

the `outputDir` must be an absolute path (no use of ~/mypath/)
- `nbCycle`: the number of cycles (iteration) that you want to use for the selfcal process. The first cycle starts at 15 times the best possible resolution, and the last finishes at the best possible resolution (determined from the longest baseline and the wavelength). The number of cycles must be between 5 and 15. If not, it will be fixed automatically to the closest one.

## Before Launching selfcal, a pre-processing step

As mentioned before, time chunks can be merged in frequency. A tool named `mergeSB.py` exists in the selfcal package to facilitate the merging process. To use it on CEP, launch it as

```
/opt/cep/selfcal/mergeSB.py
```

or type `source /home/vilchez/selfcal/selfcal_init.sh` to avoid typing the path.

On Flits, just type

```
mergedSB.py
```

Launching mergeSB.py without arguments will print some usage information:

```
MISSING Parameters
Usage: mergeSB.py --obsDir= --outputDir= --column2Merge=
```

#### mergeSB.py parameters

- `obsDir`: a directory containing ONLY Measurement Sets (MS) to merge their subbands for each time chunk. This directory must be filled by all time chunks and all subbands for each time chunks that the user want to merge. Example: If you want to process selfcal with 10 time chunks and 5 subbands, the obsDir must contains 10×5=50 MS.

Usually, one or several MSs are missing. It does not matter, NDPPP (use for merging subbands) will fill up with flagged values for missing measurements set. (Just one condition for missing subbands, the first and the last subbands for each time chunks must exist.) As for `selfcal.py`, the path of `obsDir` must be an absolute path (no use of ~/mypath/)

- `outputDir`: the output directory after merging the subbands. In this directory, two subdirectories will be created. The first one named `Merged_Data` will contain the merged time chunks (which will be use after as `obsDir` parameter for `selfcal.py`). The second one named `NDPPP_Parset` will contain the parsets that are used for merging the subbands. Inspect these parsets to find out how the subbands have been merged by `mergeSB.py`. As for selfcal.py, the path of the outputDir must be an absolute path (no use of ~/mypath/)
- `column2Merge`: the data column that you want to merge. Usually, it is `DATA` or `CORRECTED_DATA`.

### Naming scheme

`mergeSB.py` accepts intermediate or final data from the Radio Observatory, which follows specific naming conventions. So the nomenclature is specific like: * Intermediate data: `L123456_SAP123_SB123_uv.MS.dppp` * Final data: `L123456_SB123_uv.dppp.MS` If the name of your data is different, you have to change manually the name of its data using one of these nomenclature. In addition, you have to be be careful that the measurement sets will be ordered alphabetically (so use `00` instead of '0' if you have a number greater than 10 for example).

For `selfcal.py` the name of the frequency merged MS does not need to follow any convention (except that alphabetical order must correspond to the chronological order), but it will always work on the column `CORRECTED_DATA`, which is created by `mergeSB.py`.

So it is highly recommended to use these 2 tools together (except if you want to work on only one subband).

## Self-Calibration global strategy

The self-Calibration process consists in a unitary tasks, which is looped, and at each cycle (iteration) the image resolution is increased to improve the sky model, allowing a better phase calibration to be performed at the next step.
The unitary task is defined as:

- Calibration with a skymodel using calibrate-stand-alone tool (BBS)
- Flagging using NDPPP tool
- Imaging using AWimager. See the Imaging Cookbook
- Sky model extraction source extractor. The PyBDSM manual is available at http://tinyurl.com/PyBDSM-doc (html version) or ftp://ftp.strw.leidenuniv.nl/pub/rafferty/PyBDSM/PyBDSM-1.8_manual.pdf (pdf version).

The scheme below shows an example of the self-calibration process. At each cycle, each time chunk is calibrated separately using (BBS), then each of them is flagged using NDPPP. It is necessary to calibrate and flag them separately because it is not possible (at the moment) to calibrate non continuous time concatenated MS using BBS.

After this step (which is parrallelized on 8 cores), the MSs are concatenated in time using `msconcat` command from pyrap.tables module, then AWimager image the time concatenated MS with specific parameters. To finish, a sky model is extracted using pybdsm on the new image.

The goal is to increase the resolution at each cycle, that means selfcal must include more long baselines at each cycle to have a beam size coherent with the pixel size (at the moment the beam resolution is equal to 2.5 time the pixel size, but it can change in the future). Because at each cycle, the code includes more long baselines, robust parameter changes also to give weights to long baselines. It starts at robust=1 (quite natural weighting) at low resolution (15 times the best resolution) and ends robust=-2 (uniform weighting) after including all baselines available.

On the scheme, an example of the self-calibrated process with:

- 31 Time chunks
- 10 Subbands

- Best beam resolution = 5arcsec i.e best pixel resolution = 2.5arcsec

### Self-calibration starts: determination of parameters

Several checks happens when the code is launched like: number of subbands, frequency, number of channel/subbands of each time chunks etc … Each time chunks must have the same structure, if not, the code stops and write an error message in the log (or in the terminal). If checks are alright, selfcal.py will use the central frequency and the longest baseline to determine the best possible beam resolution using: Beam_Resolution = λ/Max(Baseline) ; where λ is the wavelength (calculated with the frequency).

After the determination of the best beam resolution of the data, then the best pixel size is determined by: Beam resolution = 2.5xPixel size (can be change in the future). So, the code determine the beam resolution (and the pixel size) for cycle 0 (iteration 0) by degrading it like: beam resolution cycle 0= 15x best beam resolution.

With the start and end (cycle 0 and cycle N) pixel size and beam resolution, the code estimates for each cycle its major parameters:

- beam resolution
- number of pixels
- pixel size
- robust parameter
- UVmax (use to select baselines in accordance with the beam resolution)
- wmax

After the determination of major parameters for each cycle, data will be copy in the OUTPUT_DIR/Iter0 directory and start with cycle 0.

Then target coordinates are red in the MS to generate a sky model (GSM skymodel based on NVSSS catalogue) using the gsm.py command. The GSM sky model will be used for calibration at cycle 0.

N.B: A memory criterion is computed to avoid to use too much memory. Calibration, Imaging and source extraction process have a huge memory consumption. To avoid to crash, if data are too voluminous, the best beam resolution is degraded and adapted to have a reasonable memory use. On CEP it is suggested to use (for avoiding degradation of the beam resolution) for a typical observation of 10 hours maximum and 1 subband or 5 hours 2 subband etc. This is due to the memory of lce nodes (only 16Gb). On flits, 10 hours with 20 subbands is alright (memory ~260 Gb).

### Self-calibration performances

Then the self-calibration runs and looping extracting a more complete sky model at each cycle, improving the phase calibration of the data, and the quality of the final image.
A comparison between a GSM phase calibration (cycle 0) and a complete self-calibration, at the best beam resolution, it appears an improvment of the noise about a factor between 3 to 5. And the final noise level is about 2-3 times the thermal noise.

For more complete information about parameters used, or computed inside see:self-cal in details