CMIP6 Specifications for Regridding Weights

Karl E. Taylor, Robert Oehmke, Paul Ullrich, Charles S. Zender, Sasha Ames, Paul J. Durack, Slava Kharin, Li Liu, Matthew Mizielinski, Martina Stockhause, and Sophie Valcke

5 February 2019
(Short URL: https://goo.gl/dvUgxd)

To avoid archiving CMIP6 model output fields at multiple resolutions, modeling groups can provide "weights" for use by analysts in mapping (i.e., interpolating, remapping, or regridding) model output from the grid on which it was originally reported (i.e., the "source" grid) to one of a small number of "destination" grids. The regridding is achieved by a computationally inexpensive sparse matrix multiplication of source grid values to obtain the destination grid values.

There are a number of different regridding methods commonly applied to model output, and the choice of method often depends on a number of factors including: 1) model formulation, 2) the relative resolution of the source and destination grid, 3) the variable being regridded, 4) the importance of retaining various characteristics of the original field (e.g., fundamental physical constraints and conservative properties), 5) the treatment of "missing" data, and 6) the discrete nature of some fields (e.g. when an index is used to identify various surface types). The names used to refer to some of the most common methods are "conservative", "bilinear", and "nearest neighbor". Although output may be regridded using more than one method, *it is a requirement that one of the methods must always be conservative, and this holds for each destination grid.*

Most software packages that include a regridding capability have followed the lead of the original SCRIP regridder, and so the files that contain the weights needed to perform regridding have much in common no matter which package is used. Nevertheless the files do have some differences. For CMIP6 it is required that the information needed to regrid data be saved in netCDF files with commonly structured data and metadata, and following specified controlled vocabularies for some of the metadata and all of the variable names. Here we provide specifications for CMIP6 regridding files.

Requirements and approach for recording and archiving regridding weights:

There are three major requirements regarding the files containing regridding weights;:

1. Must enable search/selection of the files through the ESGF search API and via the CMIP6 CoG, and must ensure they are retrievable from the CMIP6 archive with the same tools used to obtain CMIP6 model output files.
2. Should minimize the burden imposed on modeling groups to produce and store the weights.

3. Should minimize disruption of ESGF and other CMIP6 infrastructure, including ES-DOC, the CMIP6 CV's, Synda, and citation and errata services.

Requirement 1 will be addressed by

- imposing a directory structure and specifying filenames that are consistent with other CMIP6 files
- Requiring certain global attributes be included in each of the files containing regridding weights

Requirement 2 will be addressed by

- Adapting the ESMF regridder's weights file (which is similar to the SCRIP file), so that it satisfies requirement 1. The ESMF software is relied on by several climate data analysis tools and some "couplers".
- Suggesting that the code developers of major regridding tools augment their capabilities to output weights files consistent with CMIP6 requirements.

Requirement 3 is largely met by addressing requirement 1.

What destination grids are needed?

General guidance concerning grids for reporting CMIP6 output is provided by the CMIP Panel and the WGCM Infrastructure Panel. It must be recognized that certain analyses are best performed based on data reported on a model's native grid. Increasingly in climate models, however, native grids are not simply structured as cartesian longitude by latitude spherical coordinate systems, which complicates analysis. There are good practical reasons to enable users to easily transform data reported in non-spherical coordinates to a regular longitude by latitude grid. Another consideration is that data from different models with different grids cannot be directly compared to observations unless all model output is regridded to the grid of the observations. This leads to the recommendation that modeling groups should provide regridding weights to enable users to regrid the model output as reported in the CMIP6 archive to some common destination grids:

1. If the output is reported on a grid other than a spherical coordinate grid (i.e., as a cartesian longitude by latitude grid), then weights should be provided to map the data to a spherical coordinate grid with resolution that best represents the original data.
2. For ocean data, weights should be provided to regrid
    a. To a 1x1 degree (360x180) longitude x latitude World Ocean Atlas Grid with one of the cells centered at 0.5 E, 0.5 N, and
    b. If the grid on which the output is reported is of sufficiently high resolution, another set of weights should map data to a 0.25x0.25 degree (1440x720) longitude x latitude grid with one of the cells centered at 0.125 E, 0.125 N.

3. For all realms other than the ocean, weights should be provided to regrid to the following longitude by latitude grids, unless a model's resolution is less than that of the destination grid:

    a. 1x1 degree (360x180) longitude by latitude grid, as commonly used in reporting a variety of observations and as specified for ocean data.
    b. 2.5x2.5 (144x72), as often used in degrading the resolution of observations
    c. 0.625x0.5 (576x361), as in MERRA-2
    d. 0.75x0.75 (480x241), as commonly used by ERA-Interim
    e. 1.25x1.25 (288x145), as commonly used by ERA-40
    f. 0.25x0.25 (1440x721), for very high resolution models (for comparison with ERA5)

    For grids with an even number of latitudes (a and b above), the center of one of the grid cells should be located half a grid cell width north of the equator and half a grid cell width east of the prime meridian (and the grid cells nearest the pole will be centered half a grid cell width equatorward of the pole). For grids with an odd number of latitudes (c, d, e, and f above), one of the grid cells should be centered exactly at the prime meridian on the equator (and there will be cells of half latitudinal width surrounding each pole with latitude coordinate on the pole).

Weights may also be provided to regrid data to other spherical coordinate grids that the data provider thinks might be useful. In order to publish a weights file on ESGF, the grid must be registered at the WCRP CV registration site.

What regridding methods are acceptable?

For each destination grid defined, a set of weights should be defined ensuring "conservative" regridding. In addition, data providers may provide weights defined using any regridding method that will enable users to regrid a variable by applying the following straightforward matrix multiplication (where, for simplicity here, the source and destination fields are assumed to be defined everywhere, i.e. no "missing" values, "masking", or fractional areas):

```
! Initialize destination field
 do i=1, n_b   ! loop over all destination cells
   dst_field(i)=0.0
 enddo

 ! Apply weights to regrid field
 do i=1, n_s         ! loop over all elements of S (the weights)
   dst_field(row(i))=dst_field(row(i))+S(i)*src_field(col(i))
 enddo
```

In the above, the destination- and source-grid fields, which usually will be two-dimensional, have been “flattened” to a single dimension. The above coding comes from an ESMF document.

The following are the currently identified regridding methods that can be used in CMIP6; conservative (required), bilinear (optional), and nearest neighbor (optional). If a different regridding method is used, it must be defined by raising an issue at the WCRP CV registration site.

Care must be taken when defining the weights for conservative regridding. In particular

- The weights should be defined everywhere, even if the variables to be regridded are never defined for certain grid cells. The weights are needed to enable users to calculate the fraction of the destination cell over which a variable is defined.
- When generating the weights, direct the weight generator software to use the true grid cell areas of both the source cells and the destination cells; do not rely exclusively on the areas estimated by the regridder. (For the ESMF weight generator, do not pass masks or fractions and do provide "true cell areas" and set "--user_areas".)
- Once the weights are generated, check that the following code executes without raising an error:

```
! Initialize sums
 do i=1, n_a   ! loop over all source cells
   sum(i)=0.0
 enddo

 ! sum weighted ratio of true areas
 !      'a' is source; 'b' is destination
 do i=1, n_s         ! loop over all elements of S (the
weights)
   sum(col(i))= sum(col(i))
+S(i)*area_b(row(i))/area_a(col(i))/
 enddo

! check that sums are equal to 1
!    (within a tolerance of 1.e-6)
 imasked = 0
 do i=1, n_a   ! loop over all source cells
   if (abs(sum(i)-1.0) .gt. 1.e-6) then
     print*, 'ERROR: weighted-sum differs from 1 at grid cell'
     stop
   endif
 enddo
```

- Check that the weights actually “conserve” the quantity that is being regridded. Also check that the area of the source and destination grids match. For a sample variable, F, check the results of the following algorithm:

```
area_tolerance = ???
var_tolerance = ???
src_area = 0.0
src_sum = 0.0
! compute area-weighted sum of some source variable: src_F
do i=1, n_a   ! loop over all source cells
  src_area = src_area + area_a(i)
  src_sum = src_sum + area_a(i)*src_F(i)
enddo

dst_area = 0.0
dst_sum = 0.0
! compute area-weighted sum of the regridded variable: dst_F
do i=1, n_b   ! loop over all source cells
  dst_area = dst_area + area_b(i)
  dst_sum = dst_sum + area_b(i)*dst_F(i)
enddo

if (abs(dst_area-src_area) .gt. area_tolerance) then
   print*, 'found mismatch of global areas'
endif

if (abs(dst_sum-src_sum) .gt. var_tolerance) then
   print*, 'found mismatch of global integrals',
            'weights do not conserve!'
endif
```

What global attributes should be saved in the “weights” files?

There are two types of global attributes that should be saved in each weights file:

1) those needed by ESGF (for publishing the dataset and to populate search facets), and
2) those needed to describe the regridding method, and provide information about the source and destination grids

In serving CMIP6 data, among the global attributes needed by ESGF are activity_id, experiment_id, sub_experiment_id, variable_id, and variant_label; the complete attribute list will appear further on in this document. The regridding weights do not belong to any single activity

or experiment, but to fit into the CMIP6 system of uniquely identifying datasets, these global attributes must be defined. The table below shows how regridding files will be identified using activity, experiment, sub-experiment, variable and variant_label. Since regridding is not an "experiment" in the sense of other CMIP6 experiments, the regridding weights are placed under a separate activity. Thus, one new activity ("regrid") and at least three new "experiments" ("conservative", "bilinear", and "nearneighbor") will need to be registered at the WCRP CV registration site.

Identification of regridding files in the CMIP6 context:

activity_id	experiment_id	sub-experiment_id	variable_id*	variant_label
regrid	conservative, bilinear, nearneighbor	none	owts360x180, owts1440x720, awts360x180, awts144x72, awts576x361, awts480x24, awts288x145	usually** r1i1p1f1

* Additional variables would be added as needed to meet the needs of individual modeling groups who want to provide other regridding options.

** A p-index (other than 1) may occasionally be needed when regridding variables stored on a native grid. If some of the variables (e.g., temperature) located at the center of grid cells and others located at the edges or corners (e.g., velocity components), then different physics indexes will distinguish among the weights needed to regrid the different fields. If the physics_index (the integer following the p in the "ripf" variant label) is 1, this indicates that the native-grid variable is located at the center of a grid cell.

Under the above approach, the following attributes are needed by the ESGF publisher and must be included in every "weights" file:

```
mip_era    = "CMIP6"
activity_id = "regrid"
experiment_id = (short identifier of regridding method;
                        e.g., "conservative")
experiment = (same as experiment_id; e.g., "conservative")
sub_experiment_id = "none"
table_id = "fx"
frequency = "fx"
product = "model_output"
variant_label = (ripf identifier that applies to the *grid*, not the
            simulation ensemble member;usually "r1i1p1f1")
data_specs_version = (version of latest "data request" release);
            e.g. "01.00.30"
creation_date = (ISO 8601 date and time string,
                      e.g., "2010-03-23T05:56:23Z")
institution_id = (institution's abbreviated name, e.g., IPSL)
source_id = (model's name, e.g., IPSL-CM6A-ATM-HR)
```

```
variable_id = (indicating realm of source grid and resolution
                    of destination grid, e.g. "awts360x180”)
realm = (realm of source grid, should be one of "atmos", or "ocean", or
                "landIce", or "land")
source_type = (should be one of "AGCM" or "OGCM" or "ISM" or "LAND")
further_info_url = https://furtherinfo.es-doc.org/<mip_era>.
            <institution_id>.<source_id>.<experiment_id>.
            <sub_experiment_id>.<variant_label>
tracking_id = (tracking_id should be of the form “hdl:21.14100/<uuid>”;
            e.g., “hdl:21.14100/02d9e6d5-9467-382e-8f9b-9300a64ac3cd”)
```

In addition, the following global attributes, which provide information about the regridder, the source and destination grids, and usage guidance must be included in every “weights” file:

```
grid_label = (source grid_label, e.g., “gn”)
grid  = (source grid description; e.g. “native cube-sphere grid”)
nominal_resolution = (nominal resolution of source grid, e.g., “50 km”))
         (see Appendix 2 of the CMIP6 specifications document)
dst_grid = (destination grid description;
          e.g., "regularly-spaced lonxlat grid")
dst_grid_nominal_resolution = (nominal resolution of destination grid,
          e.g., “1x1 degree”)
           (see Appendix 2 of the CMIP6 specifications document)

title = (identity of the regridder);
         e.g., "ESMF Offline Regridding Weight Generator"
map_method = (description of regridding method);
                  e.g., "Bilinear remapping" ;
weight_generator = (the name of the regridding tool used); e.g., "ESMF"
weight_generator_version = (regridder version identifier);
                  e.g., "5.3.0 beta snapshot"
process_invoked = (the "command" or function or subroutine call(s) used
            in generating the weights);
            e.g., "ESMF_RegridWeightGen -s src.nc -d dst.nc -m conserve
      -w awts180x360_fx_CESM2_conservative_r1i1p1f1_gn.nc --netcdf4
      --user_areas"
regrid_variables = (list of variables or groups of variables that should
                    be regridded using the weights from this file)
avoid_variables = (list of variables or groups of variables that should
                    NOT be regridded using the weights from this file)
```

Note that additional global attributes that the data provider thinks could be useful may be included but are, of course, not mandatory.

What are the specifications for variables saved in the “weights” files?

Each regridding file must include the following variables (loosely following the ESMF regridder specifications; see ESMF documentation): :

n_a and **n_b:** The number of source cells and destination cells, respectively

nv_a and **nv_b:** The maximum number of corners (i.e. vertices) around a source cell and destination cell, respectively. If a cell has less than the maximum number of corners, then the remaining corner coordinates are repeats of the last valid corner's coordinates.

xc_a(n_a) and **xc_b(n_b):** The longitude coordinates of the centers of each source cell and each destination cell, respectively. The units should be “degrees_east”.

yc_a(n_a) and **yc_b(n_b):** The latitude coordinates of the centers of each source cell and each destination cell, respectively. The units should be “degrees_north”.

xv_a(n_a,nv_a) and **xv_b(n_b,nv_b):** The longitude coordinates of the corners of each source cell and each destination cell, respectively. The units should be “degrees_east”. Values should be stored counter-clockwise (looking down on the grid). If a cell has fewer than nv_(a or b) vertices, then fill unneeded values by repeating one of the corners. The ordering of the dimensions is consistent with the C language convention that the last dimension varies fastest (as adjacent memory locations are accessed).

yv_a(n_a,nv_a) and **yv_b(n_b,nv_b):** The latitude coordinates of the corners of each source cell and each destination cell, respectively. The units should be “degrees_north”. The ordering should be consistent with xv_a and xv_b, and the dimension ordering is consistent with the C language convention.

area_a(n_a) and **area_b(n_b):** The area of each source cell and each destination cell, respectively. This quantity is either from the source grid file or calculated by ESMF_RegridWeightGen. When a non-conservative regridding method (e.g. bilinear) is used, the area is set to 0.0. The units should be “m2”.

src_grid_dims(src_grid_rank) and **dst_grid_dims(dst_grid_rank):** The number of cells along each dimension of the source and destination grids, respectively. For unstructured grids this is equal to the number of cells in the grid.

n_s: The number of entries in the regridding matrix.

col(n_s): The position in the source grid for each entry in the weights matrix. Note that this pointer follows the 1-based convention for indices in Fortran. When applying in a C code, regridders must subtract one to obtain the correct entry into a 0-based S matrix.

row(n_s): The position in the destination grid for each entry in the weights matrix. Note that this pointer follows the 1-based convention for indices in Fortran. When applying in a C code, regridders must subtract one to obtain the correct entry into a 0-based S matrix.

S(n_s): The weight for each entry in the regridding matrix.

For the above variables that are not dimensionless, a **units** attribute should be included.

Should weights account for data that is undefined over portions of the grid?

Data that are “missing” or undefined over a portion of a grid (e.g. the ocean fields over land) complicate the generation and interpretation of the regridding weights. In general the masks can vary with time (e.g., a sea ice mask) or with depth (e.g., an ocean field mask). Here in order to simplify the algorithms and to ensure weights can be used for all variables reported on some source grid (both variables that are masked and those that are not), we assume that regridding weights have been generated for all grid cells (even those that might sometimes or always be masked). In certain cases a more efficient handling of fields with missing values would lead to some reduction in execution time. Although weights will be available everywhere, they may be modified, as described below, in regions where a variable is undefined over a grid cell or over a portion of a grid cell.

When a variable may be undefined on some source cells or may represent only a portion of a source cell, slightly different approaches should be taken, depending on whether the method is conservative or bilinear. The bilinear method is the simpler because the concept of a cell fraction does not apply; source values are either defined or missing. In this case the following pseudo code could perform the regridding::

Bilinear method application:

```
! input required:
     ! amissing: missing value indicator (e.g., 1.e20)
     ! n_a: number of elements in src_field
     ! n_b: number of elements in dst_field
     ! n_s: number of weights
     ! src_field(n_a): variable to be regridded
     !          should be set to amissing where undefined or masked
     ! S(n_s): bilinear weights
     ! col(n_s): pointer to source element
     ! row(n_s): pointer to destination element

! output generated:
     ! dst_field(n_b): regridded field

! Initialize destination weight accumulator
do i=1, n_b
   dst_field(i) = 0.0
   wt_sum(i) = 0.0    ! accumulator for weights
enddo

! Apply weights to regrid a src_field
do i=1, n_s         ! loop over all elements of S (the weights)
```

```
      if (abs(src_field(col(i)) - amissing) .gt. 1.e-6*abs(amissing))
           dst_field(row(i)) = dst_field(row(i)) +
                               S(i)*src_field(col(i))
            wt_sum(row(i)) = wt_sum(row(i)) + S(i)
      endif
enddo

! divide by sum of weights
do i=1, n_b
      if (wt_sum(i) .eq. 0.0)
           dst_field(i) = amissing
      else
           dst_field(i) = dst_field(i)/wt_sum(i)
      endif
enddo
```

In the case of conservative regridding, there are two options to consider: 1) preserve the area-weighted integral of a field, or 2) preserve the area-weighted global mean of the field. When there is no masking (i.e., all "frac" values are set to 1), the two options lead to the same result. For the more general case, a slightly different procedure must be followed depending on whether preserveIntegral = .true. or .false, as illustrated in the following regridding code:

Conservative method application:

```
! input required:
      ! preserveIntegral:
      !              set to .true. if the integral should be
preserved;
      !              set to .false. if the mean should be preserved
      ! amissing: missing value indicator (e.g., 1.e20)
      ! n_a: number of elements in src_field
      ! n_b: number of elements in dst_field
      ! n_s: number of weights
      ! src_frac(n_a): fraction of source cell where src_field is
      !                     defined
      ! area_a(n_a): true area of source grid cells
      ! area_b(n_b): true area of destination grid cells
      ! src_field(n_a): variable to be regridded
      !          [where equal to amissing, undefined (masked)]
      ! S(n_s): bilinear weights
      ! col(n_s): pointer to source element
      ! row(n_s): pointer to destination element

! output generated:
```

```
      ! dst_field(n_b): regridded field
      ! dst_frac(n_b): fraction of destination cell where dst_field
is
      !                 defined

!  initialize accumulators
 do i=1, n_b
     dst_field(i) = 0.0   ! accumulator for cell values
     dst_frac(i) = 0.0    ! accumulator for cell fractions
     wtsum(i) = 0.0       ! accumulator for weights
 enddo

 ! regrid field
 do i=1, n_s       ! loop over all elements of S (the weights)
   if (abs(src_field(col(i))-amissing) .gt. 1.e-6*abs(amissing))then
      dst_field(row(i)) = dst_field(row(i)) +
             S(i)*src_frac(col(i))*src_field(col(i))
      dst_frac(row(i)) = dst_frac(row(i)) + S(i)*src_frac(col(i))
   endif
   wtsum(row(i)) = wtsum(row(i)) + S(i)
 enddo

if (preserveIntegral) then
   ! preserve global integral

   !  calculate destination fractions
   do i=1, n_b
      dst_frac(i) = dst_frac(i)/wtsum(i)
   enddo

   scale = 1.0

else
    ! preserve global mean

    !  calculate total unmasked source and destination areas, and
    !  calculate destination fractions

    src_area = 0.0
    do i=1, n_a
       src_area = src_area + area_b*src_frac(i)
    enddo
    !  calculate total unmasked destination area
```

```
    dst_area = 0.0
    do i=1, n_b
       dst_frac(i) = dst_frac(i)/wtsum(i)
       dst_area = dst_area + area_b*dst_frac(i)
    enddo

   scale = dst_area/src_area

endif

! finish up
 do i=1, n_b     ! loop over all destination cells
    if (dst_frac(i) .eq. 0.0) then
        dst_field(i) = amissing
    else
         dst_field(i) = scale*dst_field(i)/dst_frac(i)
    endif
 enddo
```

Further information concerning application of the weights can be found in Taylor (2024; doi:10.5194/gmd-17-415-2024).

How should file names and directory structures be defined?

The file name and directory structure for regridding datasets must follow the template specified for all CMIP6 files:

filename template= <variable_id>_fx_<source_id>_<experiment_id>_ <variant_label>_<grid_label>.nc

Example: awts360x180_fx_IPSL-CM6A-ATM-HR_conservative_r1i1p1f1_gn.nc

directory structure template = CMIP6/regrid/<institution_id>/<source_id>/<experiment_id>/ <variant_label>/fx/<variable_id>/<grid_label>/<version>

Example: CMIP6/regrid/IPSL/IPSL-CM6A-ATM-HR/conservative/ r1i1p1f1/fx/awts360x180/gn/v20190122

What software is available to generate the regridding weights?

No currently available regridder will produce files fully consistent with the CMIP6 specifications, but the ESMF weight generator comes close. Documentation for regridders used at various modelling centres follows:

C-Coupler2: https://www.geosci-model-dev.net/11/3557/2018/gmd-11-3557-2018.pdf

CDO: https://code.mpimet.mpg.de/projects/cdo/wiki

ESMF: https://www.earthsystemcog.org/projects/esmf/regridding
http://www.earthsystemmodeling.org/esmf_releases/public/ESMF_7_1_0r/ESMF_refdoc/node3.html#SECTION03029000000000000000

NCO: https://sourceforge.net/projects/nco/
http://nco.sourceforge.net/nco.html

OASIS: https://www.geosci-model-dev.net/10/3297/2017/gmd-10-3297-2017.pdf
http://www.cerfacs.fr/oa4web/oasis3-mct_4.0/oasis3mct_UserGuide/node42.html
http://www.cerfacs.fr/oa4web/oasis3-mct_4.0/oasis3mct_UserGuide/node48.html#subsec_mapdata

SCRIP https://github.com/SCRIP-Project

TempestRemap: https://journals.ametsoc.org/doi/10.1175/MWR-D-14-00343.1
https://journals.ametsoc.org/doi/10.1175/MWR-D-15-0301.1
GitHub repo: https://github.com/ClimateGlobalChange/tempestremap

YAC: https://www.geosci-model-dev.net/9/2755/2016/gmd-9-2755-2016.pdf

Sample regridding weights file.

Variables and attributes highlighted in yellow below are required; other information is recommended but optional. [Additional variables and attributes routinely included by the code generating weights may also appear in weights files.]

```
netcdf awts360x180_fx_CESM2_conservative_r1i1p1f1_gn.nc {
dimensions:
        n_a = 82536 ;
        n_b = 64800 ;
        n_s = 24568 ;
        nv_a = 6 ;
        nv_b = 4 ;
```

```
variables:
        int src_grid_dims(src_grid_rank) ;
        int dst_grid_dims(dst_grid_rank) ;
        double yc_a(n_a) ;
               yc_a:standard_name = "latitude" ;
               yc_a:long_name = "Latitude of Source Grid Cell Centers" ;
               yc_a:units = "degrees_north" ;
               yc_a:bounds = "yv_a" ;
        double yc_b(n_b) ;
               yc_b:standard_name = "latitude" ;
               yc_b:long_name = "Latitude of Destination Grid Cell Centers" ;
               yc_b:units = "degrees_north" ;
               yc_b:bounds = "yv_b" ;
        double xc_a(n_a) ;
               xc_a:standard_name = "longitude" ;
               xc_a:long_name = "Longitude of Source Grid Cell Centers" ;
               xc_a:units = "degrees_east" ;
               xc_a:bounds = "xv_a" ;
        double xc_b(n_b) ;
               xc_b:standard_name = "longitude" ;
               xc_b:long_name = "Longitude of Destination Grid Cell Centers" ;
               xc_b:units = "degrees_east" ;
               xc_b:bounds = "xv_b" ;
        double yv_a(n_a, nv_a) ;
               yv_a:units = "degrees_north" ;
        double xv_a(n_a, nv_a) ;
               xv_a:units = "degrees_east" ;
        double yv_b(n_b, nv_b) ;
               yv_b:units = "degrees_north" ;
        double xv_b(n_b, nv_b) ;
               xv_b:units = "degrees_east" ;
        double area_a(n_a) ;
               area_a:standard_name = "cell_area" ;
               area_a:units = "m2" ;
        double area_b(n_b) ;
               area_b:standard_name = "cell_area" ;
               area_b:units = "m2" ;
        int col(n_s) ;
               col:long_name = "Pointer to Source Grid Element (with 1,
                         not 0, indicating the first element)"
        int row(n_s) ;
               row:long_name = "Pointer to Destination Grid Element" ;
        double S(n_s) ;
               S:long_name = "Weights to Map Variables from Source to
                         Destination Grid" ;

// global attributes:
```

```
:title = "ESMF Offline Regridding Weight Generator" ;
:map_method = "Bilinear remapping" ;
:weight_generator = "ESMF"
:weight_generator_version = "5.3.0 beta snapshot" ;
:history = "ESMF_RegridWeightGen -s src.nc -d dst.nc
    -m conserve -w awts360x180_fx_CESM2_conservative_r1i1p1f1_gn.nc
    --netcdf4 --user_areas" ;
:regrid_variables = "all variables except those listed in
                avoid_variables" ;
:avoid_variables = "fractions such as ocean fraction, albedo, mixing
                      ratio, etc.); also relative humidity";
:grid_label = "gn" ;
:grid  = "native cube-sphere grid" ;
:nominal_resolution = "50 km" ;
:dst_grid = "regularly-spaced lonxlat grid" ;
:dst_grid_nominal_resolution = "1x1 degree" ;
:mip_era    = "CMIP6" ;
:activity_id = "regrid" ;
:experiment_id = "conservative” ;
:experiment = "conservative” ;
:sub_experiment_id = "none" ;
:table_id = "fx" ;
:frequency = "fx" ;
:product = "model_output" ;
:variant_label = r1i1p1f1 ;
:data_specs_version = "01.00.30" ;
:creation_date = "2010-03-23T05:56:23Z" ;
:institution_id = "NCAR" ;
:source_id = "CESM2" ;
:variable_id = "awts360x180” ;
:realm = "atmos" ;
:source_type = "AGCM" ;
:further_info_url = "https://furtherinfo.es-doc.org/CMIP6.
          CESM2.conservative.none.r1i1p1f1" ;
:tracking_id = "hdl:21.14100/02d9e6d5-9467-382e-8f9b-9300a64ac3cd" ;
```