

GMI Namelist Reorganization

JULES KOUATCHOU
NASA Goddard Space Flight Center
SIVO/ASTG - Code 610.3
Greenbelt, MD 20771

April 23, 2007

Abstract

We describe how the GMI code was modified to reorganize the namelist file (renaming of the sections), to replace species indices with species labels in namelist settings, and to simplify the process of entering station information for station diagnostics.

1 Introduction

The GMI code is currently in a major transition. The code is being refactored and componentized with the objective of making it more readable, maintainable, flexible and extensible [1]. The following components have been created: Chemistry, Emission, Deposition, Diffusion, Convection and Advection. In addition, we have implemented three other supporting components: SpeciesConcentration, MetFields, and Diagnostics. With such a design, it became urgent to restructure the namelist settings in order to facilitate the componentization process.

GMI users have observed that it was difficult to set namelist variables using species indices. They would have to know (for a given mechanism) the total number of species, the position of each species in the mechanism species list, etc. Identifying species indices is a cumbersome task and is subject to errors. In addition, the task should start over when we switch mechanisms. The ideal situation would be for users to only specify the species names without worrying about the underlying species indices, and the number of species selected.

Along the same line, when doing station diagnostics, users set three different namelist variables: the total number of stations, the complete list of stations, and the location each station. The number of stations can be very large (several hundreds) and counting them, keeping track of their names and locations within the namelist file is not easy. We only need a simple way of entering the stations without being concerned of their locations and how many they are.

In this report, we explain how we reorganized the GMI namelist file by renaming sections and introducing new ones. Since there was some inconsistency in the location of some namelist variables within the namelist file, we took the initiative to move variables so that variables are grouped with respect to the component they “belong” to. We replaced

namelist variables using species indices in their setting with new ones that rely on species labels. Finally, we changed the way stations are set for station diagnostics.

2 Namelist Sections

In this section, we describe how GMI namelist sections were reorganized. This reorganization was carried out for two main reasons:

1. There were some inconsistency in the way some variables were assigned to namelist sections. For instance some diagnostics related variables did not appear in the section reserved for diagnostics (`ACTM_OUTPUT`).
2. In the process of componentizing the GMI code, the need came to restructure namelist sections so that new ones are created and that they are named according to components (Advection, Chemistry, Convection, Deposition, Diagnostics, Diffusion, Emission, MetFields).

Table 1 describes list the names of the new namelist sections.

Old Namelist Section	New Namelist Sections
ESM	ESM
ACTM_CONTROL	nlGmiControl
ACTM_INPUT	nlGmiMetFields nlGmiSpeciesConcentration
ACTM_OUTPUT	nlGmiDiagnostics
ACTM_RESTART	nlGmiRestart
ACTM_ADVEC	nlGmiAdvection
ACTM_CONVEC	nlGmiConvection
ACTM_DIFFU	nlGmiDiffusion
ACTM_DEPOS	nlGmiDeposition
ACTM_EMISS	nlGmiEmission
ACTM_CHEM	nlGmiChemistry
ACTM_PHOT	nlGmiPhotolysis
ACTM_LIGHTNING	nlGmiLightning
ACTM_TRAC	nlGmiTracer

Table 1: New namelist sections in the namelist file.

Remark 1 *It is important to note that the `ACTM_INPUT` section (mostly containing meteorological and species concentration related variables) was divided into two sections, `nlGmiMetFields` and `nlGmiSpeciesConcentrations`.*

We list in Table 2 the variables that were moved to other sections.

Namelist Variables	Old Section	New Section
pr_level_all	ACTM_CONTROL	nlGmiDiagnostics
k1r_gl	ACTM_CONTROL	nlGmiDiagnostics
k2r_gl	ACTM_CONTROL	nlGmiDiagnostics
gwet_opt	ACTM_CONTROL	nlGmiMetFields
loss_opt	ACTM_CONTROL	nlGmiChemistry
oz_eq_synoz_opt	ACTM_CONTROL	nlGmiChemistry
hdr_var_name	ACTM_INPUT	nlGmiDiagnostics
hdf_dim_name	ACTM_INPUT	nlGmiDiagnostics
lat_dim_name	ACTM_INPUT	nlGmiDiagnostics
lon_dim_name	ACTM_INPUT	nlGmiDiagnostics
prs_dim_name	ACTM_INPUT	nlGmiDiagnostics
spc_dim_name	ACTM_INPUT	nlGmiDiagnostics
rec_dim_name	ACTM_INPUT	nlGmiDiagnostics
tim_dim_name	ACTM_INPUT	nlGmiDiagnostics
AerDust_var_name	ACTM_CONTROL	nlGmiChemistry
AerDust_infile_Name	ACTM_CONTROL	nlGmiChemistry
Do_AerDust_Calc	ACTM_CONTROL	nlGmiChemistry
AerDust_Effect_opt	ACTM_CONTROL	nlGmiChemistry

Table 2: List of namelist variables that moved to other namelist sections.

The new namelist structure follows the componentization effort (still in progress) in the sense that sections are named after components or sub-components, and namelist variables are grouped according to their use.

3 From Species Indices to Species Names

Setting namelist variables using species indices is not a simple task. When moving from one experiment to another, users may be interested in a given set of species but will have to reset namelist variables matching species with their indices. In fact, their main concern is to know if a given species is part of a chemical mechanism but not its index. Mistakes may unintentionally be introduced, leading to a waste of time and computing resources. To alleviate these problems, we propose to use species labels instead of indices in the namelist file.

To achieve it, we wrote a Fortran module (*GmiSpeciesRegistry_mod* containing the function `getSpeciesIndex`) providing the species index given its name. The function `getSpeciesIndex` only takes as argument a species name and does not depend on any chemical mechanism or any particular experiment. Two variables have to be passed to the module (will do internal initialization) for the function to work properly:

- *num_species*: total number of species used in the experiment

- *const_labels*: list of all species names used in the experiment.

Remark 2 *The module gets its information at run time but not at compilation time. This allows us to take into account tracer runs without making any specific provision in the code.*

We substituted old namelist variables (using species indices) with new ones. In the namelist file, we only need to provide the list of species names we are interested in and the code will figure out (using `getSpeciesIndex`) what indices they correspond to.

In Table 3, we provide the list of the new namelist variables and the corresponding old ones. It is important to note that the old namelist variables remain part of the code (as regular variables) and are still used for calculations. The newly created variables are only local variables utilized to set the ones they replace in the namelist file.

Old Namelist variables	New Namelist Variables
nlGmiSpeciesConcentration SECTION	
fixed_const_map(1:n)	fixedConcentrationSpeciesNames
nlGmiDiagnostics SECTION	
pr_const_rec_flag(1:n)	concentrationSpeciesNames
pr_emiss_rec_flag(1:n)	surfEmissSpeciesNames
pr_drydep_rec_flag(1:n)	dryDepSpeciesNames
pr_wetdep_rec_flag(1:n)	wetDepSpeciesNames
pr_tend_rec_flag(1:n)	tendSpeciesNames
flux_species(1:n)	fluxSpeciesNames
ifreq#_species(1:n) (# = 1, 2, 3, 4)	freq#SpeciesNames
species_overpass#(1:n) (# = 1, 2)	overpass#SpeciesNames
noon_species(1:n)	noonSpeciesNames
local_species(1:n)	localSpeciesNames
col_diag_species(1:n)	colDiagSpeciesNames
nlGmiAdvection SECTION	
advec_flag(1:n)	advectedSpeciesNames
nlGmiEmission SECTION	
emiss_map(1:n)	emissionSpeciesNames
emiss_map_aero(1:n)	emissionAeroSpeciesNames
emiss_map_dust(1:n)	emissionDustSpeciesNames
nlGmiChemistry SECTION	
forc_bc_map(1:n)	forcedBcSpeciesNames

Table 3: New namelist variables used to set species names instead of species indices in the namelist file.

We adopted the following principles for the new namelist variables:

1. For a given variable, the number of species names entered is no more necessary.
2. Each variable is a long string that starts and ends with a single quote. Species names are separated with commas:
`wetDepSpeciesNames = 'H2O2, HNO3, MP, N2O5',`
3. In the previous version of the code, `emiss_map` was set in the namelist file to determine the number of species in the emission input file and to select the species to be read in from the file. In the new setting, if a species is not read in, the name to be included in `emissionSpeciesNames` is xxx. The function will return -1 for the species index.
4. The order for entering the names is not important for diagnostics related variables. However it is relevant for variables (`fixedConcentrationSpeciesNames`, `emissionSpeciesNames`, `emissionDustSpeciesNames`, `emissionAeroSpeciesNames`, `forcedBcSpeciesNames`) used to read in files.
5. Entering species names is not case sensitive. For example, the names HNO3, hNO3, HnO3, HNo3, hnO3, hNo3, Hno3, hno3 correspond to the same species. Users can select any of these names to refer to HNO3.
6. If a species name does not exist, the code will abort.

Remark 3 *In previous versions of the GMI code, it was assumed that if few species are selected for constituent, wet deposition, dry deposition, and tendency output, the first species in the mechanism will by default be included. In this work, we did not make such an arrangement. Only the species provided by the user are considered for output.*

Here is an example of namelist setting for the AURA (combo, 124 species, no ship emission) experiments:

```

dryDepSpeciesNames   = 'CH2O, H2O2, HNO3, MP, N2O5, NO2, O3, PAN, PMN, PPN, R4N2',
wetDepSpeciesNames   = 'H2O2, HNO3, MP, N2O5',
surfEmissSpeciesNames = 'CH2O, CO, NO, ALK4, C2H6, C3H8, ISOP, MEK, PRPE',
fluxSpeciesNames     = 'CH2O, CH4, CO, HNO3, H2O2, MP, NO, NO2, N2O5, O3, PAN,
SYNOZ',
freq1SpeciesNames    = 'CH4, CO, HNO3, N2O, O3, OH, C1O, C12O2, C1ONO2, HCl, CFC13,
CF2C12',
overpass1SpeciesNames = 'CH2O, CO, NO, NO2, O3, OH',
colDiagSpeciesNames  = 'CH2O, CO, HNO2, HNO3, HNO4, H2O, HO2, H2O2, NO, NO2, NO3,
N2O5, O3, OH, ALD2, ALK4, C2H6, C3H8, ISOP, PAN, PRPE, ACET',
tendSpeciesNames     = 'CH2O, CH4, CO, HNO3, H2O2, MP, NO, NO2, N2O5, O3, PAN, SYNOZ',
advectedSpeciesNames = 'CH2O, CH4, CO, H2, HCOOH, HNO2, HNO3, HNO4, H2O2, MOH, MP,

```

```
N2O, NO, NO2, NO3, N2O5, O3, Br, BrCl, BrO, BrONO2, HBr, HOBr, Cl, Cl2, ClO, Cl2O2,
ClONO2 HCl, HOCl, OClO, CH3Br, CH3Cl, CH3CCl3, CCl4, CFC13, CF2Cl2, CFC113, CFC114,
CFC115, HCFC22, HCFC141b, HCFC142b, CF2Br2, CF2ClBr, CF3Br, H24O2, ACTA, ALD2, ALK4,
C2H6, C3H8, ETP, HAC, IALD, IAP, ISOP, MACR, MEK, MVK, PAN, PMN, PRPE, R4N2, RCHO,
RCOOH, DEHYD, SYNOZ',
```

```
emissionSpeciesNames = 'xxx, xxx, NO, NO, NO, NO, NO, CO, CO, CO, MEK, MEK, MEK, PRPE,
PRPE, PRPE, C2H6, C2H6, C2H6, C3H8, C3H8, C3H8, ALK4, ALK4, ALK4, ALD2, ALD2, CH2O,
CH2O, xxx, xxx, xxx, xxx, xxx, xxx',
```

```
forcedBcSpeciesNames = 'CFC13, CF2Cl2, CFC113, CFC114, CFC115, CCl4, CH3CCl3 HCFC22,
HCFC141b, HCFC142b, CF2ClBr, CF2Br2, CF3Br, H24O2, CH3Br, CH3Cl, CH4, N2O',
```

Remark 4 *The setting of each namelist variable can be done on a single line or on several ones. In case many lines are used, it is important to begin each line on the first or second column to avoid any problem.*

We present two examples (for dry deposition diagnostics and reading the emission file) showing how the new namelist variables are used internally to set old ones.

```
!!!!!!!!!!!!!!!!!!!!!!
! For dry deposition
!!!!!!!!!!!!!!!!!!!!!!
! Set the initial value of the list
tempListNames(:) = ''

! Construct the list of names using the long string
call constructListNames(tempListNames, dryDepSpeciesNames)

num_drydep_outrecs = count(tempListNames /= '')
do ic = 1, num_drydep_outrecs
  drydep_outrec_map(ic) = getSpeciesIndex(tempListNames(ic))
end do

!!!!!!!!!!!!!!!!!!!!!!
! For emission
!!!!!!!!!!!!!!!!!!!!!!
! Set the initial value of the list
tempListNames(:) = ''

! Construct the list of names using the long string
call constructListNames(tempListNames, emissionSpeciesNames)

num_emiss = Count (tempListNames(:) /= '')
if (num_emiss > 0) then
  do ic = 1, num_emiss
    emiss_map(ic) = getSpeciesIndex(tempListNames(ic))
```

```

    end do
    num_emiss = count( emiss_map(:) > 0)
end if

```

4 Station Diagnostics

As we mentioned in the introduction section, when we want to do station diagnostics we need to set in the namelist file three variables :

col_diag_num: total number of stations

col_diag_site: complete list of stations

col_diag_lat_lon: locations (latitude/longitude) of stations.

Users need not only the count the number of stations (can be several hundreds) but also match the name of each station with its location. Removing/adding one station from the list requires the resetting of the above three variables with the possibility of mis-matching. To facilitate the selection of stations, we propose to do the following:

- Construct a file containing a list of all the possible stations and their locations. A namelist variable will be created to point to the file. New stations can be added to the file at any time. The order of writing station information is irrelevant.
- Create a namelist variable (long string) to enter the list of selected stations.
- Allow the code to check if each selected station exists in the file and extract its location.
- Allow the code to compute the number of selected stations.

To achieve it, we created two new namelist variables (see Table 4).

Old Namelist variables	New Namelist Variables
nlGmiDiagnostics SECTION	
col_diag_num	N/A
col_diag_site()	colDiagStationsNames
col_diag_lat_lon(2,n)	N/A
N/A	stationsInputFileName

Table 4: New namelist variables for station diagnostics.

The piece of code used to carry out the above operations is:

```

! Construct the list of station using the long string
call constructListNames(col_diag_site, colDiagStationsNames)

```

```

col_diag_num = Count (col_diag_site(:) /= '')

if (col_diag_num /= 0) then

  do ic = 1, col_diag_num
    ! For each station in the list, check if it exists in the file
    ! and get its position (lat/lon)
    call getStationPosition(col_diag_site(ic), &
                           col_diag_lat_lon(1,ic), &
                           col_diag_lat_lon(2,ic), &
                           stationsInputFileName)
  end do
  .
  .
  .
end if

```

We present below a sample namelist setting for `colDiagStationsNames` and the first few lines of the file (`colDiagStationList.asc`) containing station information.

```

colDiagStationsNames = 'SPO, MCM, HBA, FOR, NEU, SYO, PSA, MAR, MAQ,
TDF, CRZ, LAU, CGO, ASP, CPT, EIC, JOH, REU, NAM, FIJ, TAH, CUI, SMO,
PNA, WAT, ASC, NAT, SEY, BRA, MAL, NAI, SNC, CHR, KCO, PAR, TVD, PAN,
VEN, RPB, GMI, POO, KUM, MLO, GTK, HON, TAI, ASK',

# -----
# Station name      Lat      Lon      Description
# -----
SPO                 -89.98   335.20
MCM                 -77.83   166.60
HBA                 -75.56   333.50
FOR                 -71.00    12.00
NEU                 -71.00   352.00
SYO                 -69.00    39.58
PSA                 -64.92   296.00

```

Remark 5 *While editing the file containing the station information, the following rules apply:*

1. *The first three lines of the file should start with the # character.*
2. *Column 1 to Column 16 are for the station name.*
3. *Column 17 to Column 25 are for the latitude of the station.*
4. *Column 26 to Column 34 are for the longitude of the station.*
5. *The remaining columns are reserved to describe stations and are not read in.*

5 Conclusions and Future Work

We described the changes we made in the GMI code to reorganize the namelist file and to make it more consistent with the GMI components. In addition, we presented how the code was modified to be able to replace species indices with species names in the namelist file, and how we simplified the setting of station diagnostics variables. Our initial tests were successful and validated the code changes.

Throughout the code, species indices are passed to components, extending the argument list of component interface routines. With the modifications done in the code, we can now obtain a species index by using its name. We plan to make necessary changes so that species indices are derived within components only.

References

- [1] J. Kouatchou, T. Clune, H. Oloso, S. Zhou, M. Damon, and B. Womack, Refactoring and componentizing legacy codes: GMI case study, in preparation.