Replacing Fortran Namelists with JSON

Tom Robinson
Engility/NOAA GFDL
thomas.robinson@noaa.gov

IN41C-0051

INTRODUCTION

Scientists need a way to change their experiments without changing the model code. Namelists are currently used as key/value pair input in the GFDL climate model that change on an experiment basis. Some downfalls of namelists are

- Only single errors can be detected
- Output must be written out by the programmer
- There are no tools to check namelists
- Namelists are only part of the Fortran standard
- Tracking namelist and default value changes is difficult

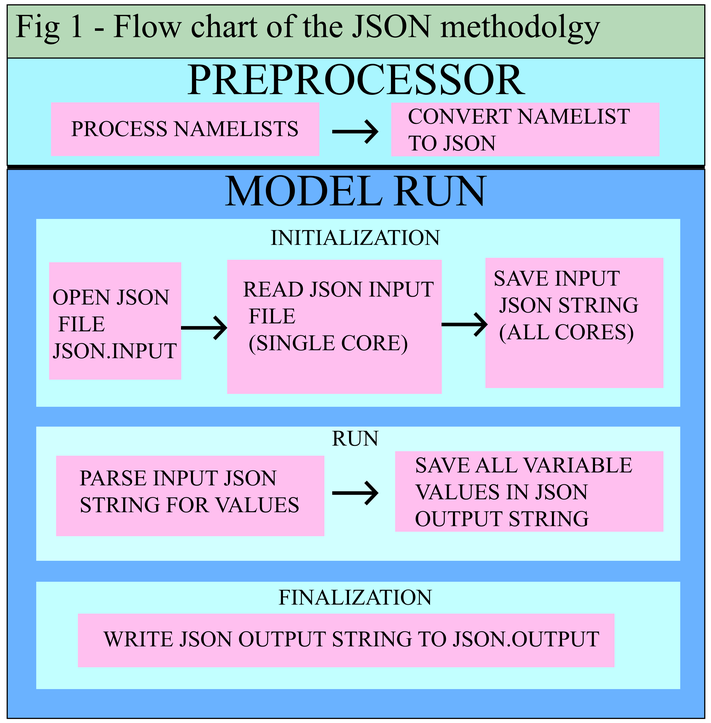
MOTIVATION

There are 214 namelist reads, but 210 writes in the GFDL Atmosphere Model 4 code (Table 1). Tracking every namelist and default value is nearly impossible. Now that Fortran is interoperable with C, we will be using C for sections of our code that are best written in that language. We also need a way to quickly check the input files for differences and figure out the owner of such changes. Additionally, output needs to be created that can easily be checked and compared to previous runs to make sure the model is using the same inputs. In order to solve all of these issues, a set of routines that process JSON have been created.

Table 1 - Number of namelist Reads, Writes, and the difference between them

Model Component	Namelists Read	Namelist Written	R - W
AtmDynamics	14	13	1
AtmPhysics	106	112	-6
Ice	6	8	-2
Ocean	15	24	-9
Land	26	26	0
Coupler	3	3	0
Infrastructure	44	24	20
total	214	210	4

Fig 1 - Flow chart of the JSON methodolgy



METHODS

A JSON is generated from an existing namelist by converting the namelist into a JSON array of JSON objects (Fig 2). The JSON is then read by the model and stored as a string. The JSON is parsed for the "namelist" and each key in the namelist. The values of the keys are extracted. Whether or not the key is in the JSON, it's value is stored in the output JSON. The output JSON is written at the model's end.

Fig 2 - Example namelist to JSON conversion

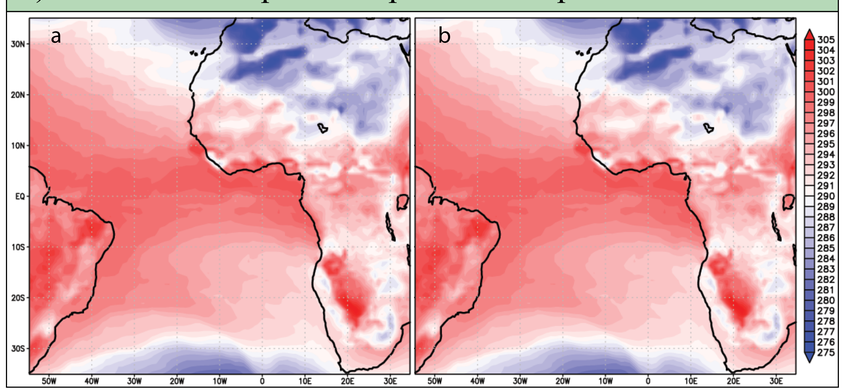
```
&example_nml
   array = 1,2,3,4,5
   array(2) = 20
   array2 = 3*20
   string='hello'
 /
```

```
{" example_nml":[
   {"array":[1,2,3,4,5]} ,
   {"array(2)":20},
   {"array2":[20,20,20]},
   {"string": "hello"}
 ] }
```

RESULTS

Because the model is reading/writing the JSON input/output only one time each, the model run time differences are not noticeable using the JSON routines. Every namelist that is read using the wrapper is written out, so there is no doubt about which values and namelists were used in the experiment. The output JSON file can be easily parsed, whereas the namelist output is written to a logfile that can contain more than just namelists. The answers bitwise reproduce the answers obtained using the namelist (Fig 3). C-functions that require user-defined input have a routine that they can call instead of calling a Fortran function to read the namelist. Errors are more descriptive as opposed to the general message used for namelists, and the JSON output file can be dumped out to see how far along the JSON reading was when the error occurred.

Fig 3 - Surface temperatures (K) using a) namelists and b) JSON as the experiment parameter input method



CONCLUSION

Replacing namelists with JSON maintains the functionality of the namelist while adding improvements such as portability, multilanguage support, and quality control checking of the input.