Package Variables: Requirements and Design

MPAS Development Team

October 18, 2013

Contents

Chapter 1

Summary

This document introduces package variables, and describes requirements and design specifications for the implementation and use of package variables.

Package variables can most easily be explained through the concept of optional physics packages. For example, one simulation might have physics package A on while the next might have physics package A off. During a simulation we might not want to have all variables allocated for this physics package when it's not being used.

As such, package variables are introduced. These are groupings of variables whos allocation depends on the choices of namelist options.

Chapter 2

Requirements

2.1 Requirement: Variable allocation depends on namelist parameter

Date last modified: 10/03/2013
Contributors: (Doug Jacobsen, Michael Duda, Todd Ringler)

Variables and variable arrays defined in registry can have their allocation controlled through a run-time namelist parameter.

Chapter 3

Design and Implementation

3.1 Implementation: Package Variables

Date last modified: 10/03/2013
Contributors: (Doug Jacobsen)

Package variables should be defined in registry. First a namelist option needs to be defined that will control the package but could more generally represent the presence of an optional physics package or portion of software.

As an example, we will say config_physics_a is the namelist option we're interested in, and we will assume this namelist option has already been defined in registry correctly.

The first addition to Registry.xml will be the option to have:

```
persistence="package"
```

This option is added to the var_struct, var_array, and var constructs. But not to the var construct nested under a var_array, since individual constituents can't be allocated within a var_array.

When

```
persistence="package"
```

and two additional attributes are added to each of the constructs.

```
package_key="config_physics_a"
package_value=".true."
```

where config_physics_a is the "package key" that controls the allocation of the construct.

Within the registry code, additional logic will be added to check if a construct is defined as "package" or not. If it is, then additional Fortran code will be added to the allocations and deallocations that are controlled by the namelist parameter supplied in the package attribute.

When persistence, package_key, and/or package_value are set on a var_struct, they define the default attribute values for all var and var_array constructs within the var_struct. These can be overwritten be adding persistence, package_key, and/or package_value to the individal var and var_array constructs.

Conditional logic for using these package variables needs to be controlled by the actual core defining/using them.

When package_key is specified, package_value needs to be specified as well. If it is not specified, registry will error and fail to generate code or build the model.

Within registry, when the parser sees a package_key attribute, it searches through all namelist options to find matching one. After the matching one is found, the variable/group/variable array is linked to that namelist option. When the Fortran code is being generated to allocate the variable, an if test is added around the allocation that checks if the namelist options value is equal to the package value attribute. This comparison is different depending on the type of the namelist option. The linking described earlier occurs to allow easy checking of the type of the namelist option to cause the comparison to happen correctly.

The package_value attribute is allowed to be a semicolon delimited list of values. If the list contains multiple values, the logic generated by registry allows the variable to be allocated if the package_key is equal to any of the specified package_values. For example:

```
package_key="config_physics_a"
package_value="phys1;phys2;phys3"
```

Would generate logic similar to:

```
if(config_physics_a == trim('phys1') .or. &
   config_physics_a == trim('phys2') .or. &
   config_physics_a == trim('phys3') ) then
     allocate(var)...
end if
```

Chapter 4

Testing

4.1 Testing and Validation: Package Variables

Date last modified: 10/03/2013
Contributors: (Doug Jacobsen)

Package variables will be added to a component.

A run with the package on and off will be performed, and then should both run to completion and produce bit-idential results to runs where the package variables are defined as persistent.