**How to Add a Variable in UPGM in AgES-W**

**This is for all FORTRAN changes**.

1. **jupgm.f90** Path: AgES-W\Sources\upgm.
   1. **Information**
      1. In the Subroutine argument list: the first two variables are arrays: ints and reals. They hold variables that are of that data type, i.e., integers or real numbers.
      2. Then there are six special length integers that are used to dimension the sizes of arrays. New ‘len’ variables need to added to the Subroutine argument list.
      3. Arrays are only added to the argument list of jupgm. Scalars DO NOT need to be added to this list.
      4. Strings require two parameters: the name of the string and the length of the string. For example, “seedbed1,seedbed1\_len”. This is only true in Fortran and thus is required in this file.
2. **Steps**
   * 1. Add a new variable **to jupgm’s Subroutine argument list**. Do not add a scalar to this list!
     2. When adding an **int or real**, change the **DIMENSION** of the ‘ints’ or ‘reals’ array. These two arrays are arrays of scalar variables. A scalar variable holds a single value.

! @Description("Array of real arguments")

! @In

! @Out

REAL(C\_FLOAT), DIMENSION(**144**), TARGET :: reals

* + 1. To add a new **array** variable, add it to the **DIMENSION** lists of variables. This sets the size of the array. So far, all arrays are one-dimensional. A new ‘len’ variable would be added as an array in this section. NOTE: The listing of len variables is in preupgm.java. Example:

! @Description("GDDs for Germination at Soil Moisture")

! @In

REAL(C\_FLOAT), DIMENSION(lenb) :: germgdd

* + 1. **Declare the new variable**. Add a **POINTER** to it. This is for Scalar variables. Put an integer variable under the ints comment and a real variable under the real comment. Example:

! @Description("Leaf Number")

REAL(C\_FLOAT), POINTER :: ln

* + 1. Add the new variable to the **list of pointers** either under the Integer or the Real section For Example:

ln => reals(144) This shows the position in the ‘reals’ array.

* + 1. To **print** out the value of the variable before jupgm executes the calls to subroutines etc. add the new variable to the list under the **printflg section**. An example: ‘PRINT \*, 'leaf number:', ln ‘
    2. Add the **new variable to the subroutine’s arguments list** that uses the new variable and is called from within jupgm.f90.This is how the new variable is passed to the Fortran subroutine**.** For Example:

CALL callcrop(aepa,aifs,am0jd-plant\_jday+1,1,antes,antss,blstrs,boots,browns, &

& callgdd,canht,canopyflg,cliname,cots,cropname,dayhtinc,dents, &

……

& end\_stem\_mass,ln)

* + 1. To **print** the new variable’s value AFTER the program has executed, add the new variable to the list of **PRINT** statements at the end of the file. For Example:

PRINT \*, 'ln:', ln

**The following is for all JAVA changes.**

1. **HRU\_java** Path: AgES-W\Sources\ages.types\HRU\_java This class simply declares all the variables. **A ‘real’ in** Fortran is a ‘float’ in Java. ‘Integers’ in Fortran are ‘integers’ in Java.
   1. **Declare the variable** in the UPGM section, i.e., in the \\UPGM commented section. Include the description too. The ‘@’ is an annotation symbol. For Example:

@Description("Leaf Number")

public float ln;

* 1. **Declare an array** here too. Example:

@Description("Seedling Elongation Rate at Soil Moisture")

public float[] ergdd;

* 1. **Declare strings** like arrays but WITHOUT the length variable. The length is not needed in Java.

@Description("Name of crop currently on HRU")

public String cropName;

1. **Upgm.java** Path: AgES-W\Sources\upgm. This class shows the flow of variables.
   1. Three classes are called from Upgm.java. They are run sequentially and include:
      1. PreUpgm
      2. JupgmGen
      3. PostUpgm
   2. **Flow:**

**HRU.java -> PreUpgm -> Jupgm ->PostUpgm -> HRU.java**

field2in out2in out2in out2field

1. All variables occur in the First Transfer from **HRU.java to PreUpgm**. This utilizes **‘field2in’**. Examples:
   1. Arrays: field2in(hru,"germgdd",preupgm);
   2. Strings: field2in(hru,"acynmu",preupgm);
   3. Scalars: field2in(hru,"ln",preupgm);
2. The Second Transfer from **PreUpgm to Jupgm** utilizes **‘out2in’**. This is only necessary for Arrays and Strings. Examples:
   1. Arrays: out2in(preupgm, "germgdd", jupgm);
   2. Strings: out2in(preupgm, "acynmu", jupgm);
3. The Third Transfer from **Jupgm to PostUpgm** utilizes **‘out2in’**. Again, this is only necessary for Arrays and Strings. Examples:
   1. Arrays: out2in(jupgm, "aifs", postupgm);
   2. Strings: NO EXAMPLE (same format as arrays)
4. The Fourth Transfer goes from **PostUpgm to HRU.java** and utilizes **‘out2field’**. This is for all variables. Examples:
   1. Arrays: out2field(postupgm,"aifs",hru);
   2. Strings: NO EXAMPLE (same format as arrays)
   3. Scalars: out2field(postupgm,"ln",hru);
5. **PreUpgm.java**. Path: AgES-W\Sources\upgm.
   1. Annotation symbols are used in these files and are necessary to communicate with OMS. Annotation symbols are denoted by ‘@’. Using ‘Annotations’ gives more description to the variable.
   2. **The** **variables are listed with their annotations**. Add the new variable with the appropriate annotations such as ‘@Description’ or ‘@Unit’ etc. Examples:
      1. Arrays: @Description("Seedling Elongation Rate at Soil Moisture")

@Unit("m/d")

@In @Out public float[] ergdd;

Arrays require both ‘@in’ and ‘@out’.

* + 1. Strings: @Description("Soil Moisture Condition of Seedbed")

@In @Out public String acynmu;

* + 1. Scalars: @Description("Leaf Number")

@In public float ln;

1. The ‘len’ variables are also listed here. For Example: @Out public int lena;. The two serialized arrays for ‘ints’ and ‘reals’ are listed here too.
2. Serialize means to put variables into one object.
3. NOTE: There are additional UPGM variables listed which are grouped together with like variables and are under a comment indicating this. For example: // UPGM stress variables.
4. The execute method follows next. The ‘len’ variables are listed here and given a value for each one of the six. It is in this section that variables are defined.
5. In the **init method**, add the new variable and give it an initial value. Examples:
   * 1. Arrays: Initialize the size of the array. Example:

egdd = new float[lena];

antss = new int[]{999,0,0,0};

1. Scalars: ln = 0f;
   * + 1. 0f = float
       2. 0.0 = double
       3. 0 = integer assumed
2. **JupgmGen.java** Path: AgES-W\Sources\upgm. **Only Arrays or Strings** need to be added to this file. This class runs the Fortran code and copies the dll to UPGM.
   1. **Define the array or string.** Examples:
      1. Arrays: @Description("JulianDay/YYYY/MM/DD of Sunflower Head Yellowing")

@In

@Out

public int[] yelows;

NOTE: Don’t pass the array ‘out’ if UPGM does not use it.

* + 1. Strings: @Description("Name of Crop")

@In

public String ac0nam1;

1. If a new ‘len’ variable needs to be added, this is the place to define it. Example:

@In

public int lenf;

1. The lib.jupgm calls the Fortran code. The parameter list needs to be updated to match the Fortran subroutine. This means the variable needs to be added to the list in the same place as the Fortran code. Integers do not need to be added in JupgmGen.
2. **PostUpgm.java** Path: AgES-W\Sources\upgm.
   1. **Define arrays and Variables.** 
      1. Define Arrays as ‘@In’ and’@Out’. Example:

@Description("Germination GDD Values Plus Values for 2 Intermediate Soil Moisture Levels")

@In @Out public float[] ggdd;

* + 1. Define Scalars as ‘@out’. Example:

@Description("Leaf Number")

@Out public float ln;

1. In the **execute** **method** under the ‘\\Integer Deserialization’ or under the ‘\\Real Deserialization’ commented sections, enter the new variable in the proper section. This lists the variable and the array position. This is done to deserialize the ‘ints’ and ‘reals’ arrays. Examples:

acdayspring = ints[31];

ln = reals[143];

1. Under the ‘ints’ section, the first variable listed is in the 10th array position. There are other variables in positions 0 -9 which are temporary and do not need to be saved.
2. NOTE: Arrays in Java start with position 0 (zero).
3. After the new variable has been added as described above, then test the addition by compiling AgES-W and running it. Then look at the output to see if the values are appropriate. To view the output, the new variable must be added to the ‘output.inc’ file in the project for which you are running and testing the new variable.
   1. To do this go to the file ‘output.inc’ in the project and open it.
   2. For Crop variables, go to the line: ‘attrSet\_crop\_upgm and add the variable at the end of the listing. Example:
      1. attrSet\_crop\_upgm "ID;acmstandstem;acmstandleaf; …. temp\_sai;trad\_lai;ln"