## UMUI Jobs in swsvalde

## Introduction

When I started using the UM, I inherited a lot of different jobs from the Hadley Centre. These jobs covered the whole range of scientifically different versions of the model, from normal HadAM3-N48 (and HadAM3-N216), to HadSM3 and HadCM3-M1, and HadCM3-M2.1. However, the jobs often had lots of differences, due to different users, differences in STASH, the date when the job was created, different machines etc. etc. This made it very difficult to compare jobs and work out what was a true scientific difference, and what was just a path or machine technical difference. I also wanted to be able to submit to a range of different machines without having to make lots of changes to the job.

## umui job structure aim

I therefore decided to create a set of jobs covering all of the major variants of the model, which:

1. When taking differences between model variants, the only differences were related to science.
2. Standard modsets were all located together so path names were simple and clear.
3. Standard naming of folders, using variable names for key locations, so that moving to different folder names on different machines requires just one simple change.

Moreover, I then implemented a script, clustersubmit (an extension of the UM provided umsubmit) which will automatically make the changes required for different machines, change the job from compile to run mode (all of our standard jobs are set to compile and stop), and also allows the user to select the queue, core numbers of NS and EW, and the run time.

## Overall Structure

In the umui on puma, the user called swsvalde holds a series of experiments with different configurations of the model. Each experiment will have a number of jobs, with the highest letter representing the latest, most up-to-date version. When available, the original versions were successfully benchmarked against a “standard” from the Hadley centre. E.g., the HadCM3-M1 job tcsyb was run and output compared to a Hadley Centre equivalent. HOWEVER, subsequent versions may have made small changes to the science (and bug fixed errors) so that the latest versions may not be 100% scientifically identical to the original. If you are interested, you can go through the differences to help ascertain the changes. It is on my long to-do list, to document the changes but this will probably never happen!

Within each job, there is a standard structure. The path names etc use a standard naming convention based on some system variables, described below:

## Key “System” Variables

When a job is submitted, the scripts expect a number of shell variables to be defined. This is achieved through a script called **setvars** or **setvars\_4.5** or equivalent. On our BRIDGE machines, a user should include invoke this in their .profile file. Although we have also added a few extra definitions. In particular, we define a variable called DUMP2HOLD which is the folder which contains the job folder, with the input, output, and code folders (the name originates from an old national HPC machine which used this name). This folder needs to have sufficient space to handle the output from the job.

Within a umui job, go to **Sub-Model Independent -> Script Inserts and Modifications**. There will be a further number of “Defined Environment Variables”. These are rarely (never) changed for a simulation.

|  |  |  |
| --- | --- | --- |
| Name | Description | Typical Value |
| ANCIL\_ROOT | This is the root folder for most of the system wide ancil files (note that the precise location is defined below in the File and Directory naming section) | /home/swsvalde |
| PV\_ROOT | This is the root folder for most of the system wide modset files. (Note that the precise location is defined below in the File and Directory naming section) | /home/swsvalde |
| UM\_INPUT | Folder for various standard input files | $UMDIR/PUM\_Input/vn4.5 |
| UM\_OUTPUT | Folder for various output files. I don’t think this is used! | $HOME/PUM\_Output/vn4.5 |
| TEMP | Temporary folder for a variety of files whilst model is running | $DUMP2HOLD/um/$RUNID/tmp |
| IN | Temporary input file folder. | $TEMP |

In the submission script (clustersubmit), the values for ANCIL\_ROOT and PV\_ROOT are changed to reflect the full pathname for the appropriate machine.

In addition, there are some further variables defined in: Sub-Model Independent -> **File & Directory Naming. Time Convention & Environment**. Note that these variables point towards various folders, but the variables can be ignored subsequently. For instance, in the ancil section, you can completely ignore the variables defined below and give the folder location as something completely different. This is OK to do (provided you keep to the structures e.g., using $HOME or ~username) but it is perhaps a little bit less tidy/structured than using the variables below.

|  |  |  |
| --- | --- | --- |
| Name | Description | Example Value |
| MY\_ANCIL | This should be the folder location for most of the ancil files for your job | $HOME/ancil/scotese/160\_4\_1deg  (See point 1 below). User can change to anything they want. |
| ALT\_ANCIL | This can be used if a few of your ancil files or coming from a different folder. | $ANCIL\_ROOT/ancil/preind2  (But might not be used). User can change to anything they want. |
| OZONE\_ANCIL | This is used for the folder name for the ozone file. For the vast majority of our runs, we do not change ozone. Previously we needed to copy the ozone file but now we can simply give the ozone file as our pre-industrial standard | $ANCIL\_ROOT/ancil/preind2 User can change to anything they want. |
| MY\_UPDATES | Location of your own modsets | $HOME/um\_updates  (See point 1 below). User can change to anything they want. |
| MY\_DUMPS | Location of your dumps. This is not always used. It depends on the setting in the reconfiguration. It is useful when you are using the same dump file for many simulations. | $HOME/dumps  (See point 1 below) User can change to anything they want. |
| MY\_EXECS | Location of a standard executable. Again, this is only commonly used if using the same executable for lots of simulations. | $HOME/execs  (See point 1 below) User can change to anything they want. |
| PV\_UPDATES | This is the folder location for standard updates created (or maintained) by BRIDGE | $PV\_ROOT/um\_updates  (Rarely changed) |
| PV\_DUMPS | This is the folder location for standard dump files created (or maintained) by BRIDGE | $PV\_ROOT/dumps (Rarely changed) |
| PV\_EXECS | This is the folder location for standard executables created (or maintained) by BRIDGE. Rarely used for the main model but the reconfiguration executables are stored here as they rarely change. | $PV\_ROOT/execs (Rarely changed) |
| UM\_SPECTRAL | File for radiation spectral files. Rarely changes. | $UMDIR/vn$VN/ctldata  (Rarely changed) |
| MODS | Modsets from the original UM distribution | $UM\_INPUT/mods/source (Rarely changed) |
| MODS\_SCRIPTS | Script modsets from the original UM distribution | $UM\_INPUT/mods/scripts (Rarely changed) |
| MODS\_SOURCE | Further modsets from the original UM distribution | $UM\_INPUT/mods/source/clim (Rarely changed) |
| OVERRIDES | Folder for compiler overrides | $UM\_INPUT/overrides (Rarely changed) |
| FAMOUS\_MODS | Folder for FAMOUS modsets. This may also be used in HadCM3 setup | $UMDIR/../famous (Rarely changed) |

1. If the variables are specified as $HOME, this will work for the user but will not work if another user copies your folder without them also copying the folder structure and contents. It is therefore better to give the path as ~username/ancil/scotese/160\_4\_1deg. This would work for any user without having to create a copy of the contents.
2. In addition to the variables above, we always use the following:
   1. DATAM = $DUMP2HOLD/um/$RUNID/datam
   2. DATAW= $DUMP2HOLD/um/$RUNID/dataw
   3. UCOMPDIR= $DUMP2HOLD/um/$RUNID/code

## Modsets

All non-experimental modsets which are in addition (or have been changed from the standard release) are stored in a central location $PV\_ROOT/um\_updates. In many cases, over the years, I have received multiple copies of the same name modset. In about 2010, I compared all copies of the same file and found no substantive scientific differences. A few same named modsets had differences in the comments but nothing else. However, many also included unnecessary blank characters at the end of each line, or the modset comments, which extended beyond row 72. This caused warning messages in the nupdate command which could obscure true problems. Similarly, some modsets added lines to the same location which again generated warning messages. Most were unimportant but there was an example (liked to adding water isotopes) where it could make a scientific change. Hence, I have processed all of the files in $PV\_ROOT/um\_updates so that they do not give any warnings in normal use. This means that they are probably not the same as any other version/copies that are around, but they should be scientifically identical (except for the isotope example above). If you do see a warning, you need to check it!

There are a few modsets that need to be different on different machines. The main one is a script modset to allow the submission of jobs to work. This will often be different on different machines because the submission method varies. he way we get around this is that if any modset in the umui has *local* in the name, then this means that it will be a machine dependent modset. In most cases, we normally make the actual file a symbolic link to the unique file on the machine itself. For instance, in the umui jobs, there is file called $PV\_UPDATES/mpirun-local. If you logon to Bristol’s HPC machines then the file exists but is a symlink to mpirun-bc4 on bluecrystalp4, and mpirun-bp1 on bluepebble.

## STASH

STASH output was standardised so that all versions of the model use the same STASH unless it is version specific (i.e., MOSES2.1/TRIFFID has additional outputs compared to MOSES1). The selected STASH is significantly smaller than in most Hadley Centre jobs and doesn’t use any climate meaning options. The reason I removed this is that I found it to be temperamental when there were disk space/quota problems which was common in early years and is occasionally still an issue currently.

The only exception to common STASH is with the high-resolution versions of HadAM3. The main point of using the high-resolution model is to look at extremes and hence we wish to have much larger volumes of output.

## Further changes

There are several options in the model where you can choose the fast version of the code, or one that is bit reproducible. They state that they are scientifically identical, and I assume this is the case. Jobs inherited from the Hadley Centre were very variable in the usage.

Until recently, we always selected the fast version, but I have recently discovered (2021) that the bit reproducible version is not slower (or in one case faster)! Hence, they are now all changed to bit reproducible except in one location because the isotope code was not coded for this option.

## Summary

With these changes adopted,