

AusFarm – New Features

Version (1.4.8-) 1.4.13



1	Manager Scripts	5
1.1	Resize arrays	5
1.2	New pos() Function	5
1.3	New datewithin() Function	6
1.4	Listing Crop Cultivars.....	6
1.5	Model tree shows sequence	6
1.6	Reformatting Management scripts.....	7
2	Output variables.....	7
2.1	TextOut component.....	7
2.2	Selecting output variables.....	8
3	Component comparison	9
4	Manager scripts	10
4.1	Multi-line comments	10
4.2	New DayOfYear() function	10
4.3	Constants.....	11
4.3.1	Used for initialisation.....	11
4.4	Expressions can be used for initialisation	11
4.5	Bookmarks	12
4.6	Code completion	13
4.7	Matching braces	13
5	Initial property and parameter sections for APSIM components.....	14
5.1	Custom editor for APSIM components	14
5.2	APSIM Component Initial Values	14
6	Translator initialisation dialog	16
7	Using APSoil soil data	16

Using AusFarm's new features.

1 Manager Scripts

1.1 Resize arrays

Arrays can now be resized dynamically at runtime. This makes the scripting much more flexible and compact. Below is a simple example.

```
1
2
3 define crop_names = ['wheat', 'barley', 'canola']
4
5 on start
6 {
7     set crop_names = resize(crop_names, 5)
8
9     set crop_names[4] = 'oats'
10    set crop_names[5] = 'fieldpea'
11 }
12
```

Other dimensions of arrays can also be resized.

```
263 ! resize the template array to the size of the rotation array items
264 for i = 1 to length(rotation_template)
265     rotation_template[i] = resize(rotation_template[i], length(rot_system_info[i]:sequence))
266
```

Another technique for resizing or initialising arrays is using a method similar to what Python does. In the example below a two dimensional array is defined with initial values of '-'.

```
define dasharray = [['-'] * 3] * 2
```

Is the same as:

```
define dasharray = [['-', '-', '-'], ['-', '-', '-']]
```

Using a set during execution can be used to reinitialise or resize the array.

```
set paddocknames = ['-'] * COUNT
```

Where *COUNT* is always a constant.

1.2 New pos() Function

The pos() function finds elements in arrays or strings. The examples below show how it is used.

```

define text array[3] = ['one', 'two', 'three']
set p = pos('three', array)    ! 3

define double darray[3] = [0.1, 0.11, 0.111]
set p = pos('0.1', darray)    ! 1

define text astring = 'lo!g text form'
set p = pos(1, astring)       ! 3

define integer iarray[3] = [1, 11, 111]
set p = pos(11, iarray)       ! 2

```

All of these examples return an index value. The first parameter is converted to the type of the elements in the array.

When an item is not found a 0 is returned.

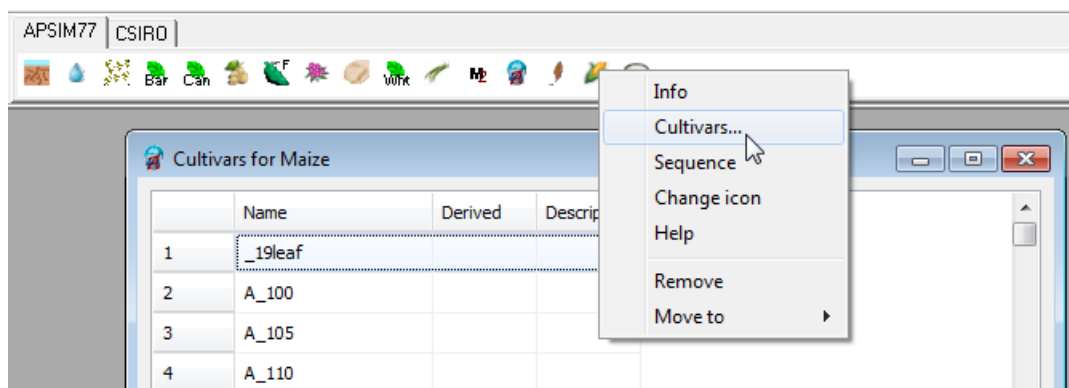
1.3 New datewithin() Function

Datewithin() is a new function that simplifies scripts where date ranges are tested. It takes into account the case where date *istart* may be later in the year than *iend*. This is when 1 Jan is in the period.

```
set inperiod = datewithin(day, istart, iend)
```

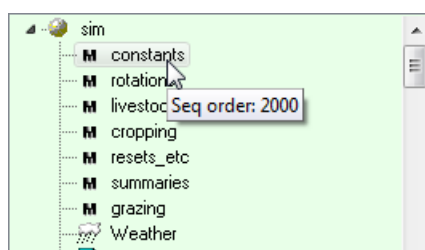
1.4 Listing Crop Cultivars

APSIM crop components can now list the cultivars by right clicking on the icon in the model palette and choosing **Cultivars**. The cultivar names can be copied to the clipboard by right clicking on the cultivar list.



1.5 Model tree shows sequence

The simulation model tree now shows clearly the execution sequence of the Management scripts.

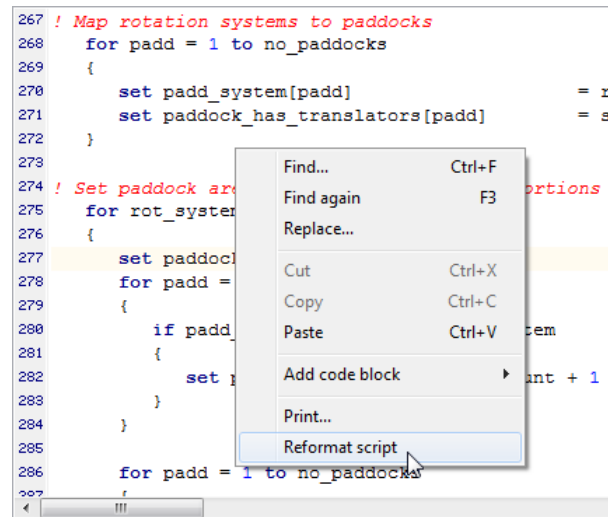


The sequence steps in the simulation run from 0 to 9999 in each daily timestep. Each component can have its events configured to run at any of these points in the daily timestep.

1.6 Reformatting Management scripts

Manager scripts can often get untidy and loss of indentation can make them difficult to read. A new option to reformat the script based on some rules is now available when you right click on the script editor. When the script is reformatted, the curly braces will appear on their own lines. Indentation will be adjusted with indents of three characters per indent. Comments will not be adjusted unless they are on the end of a line.

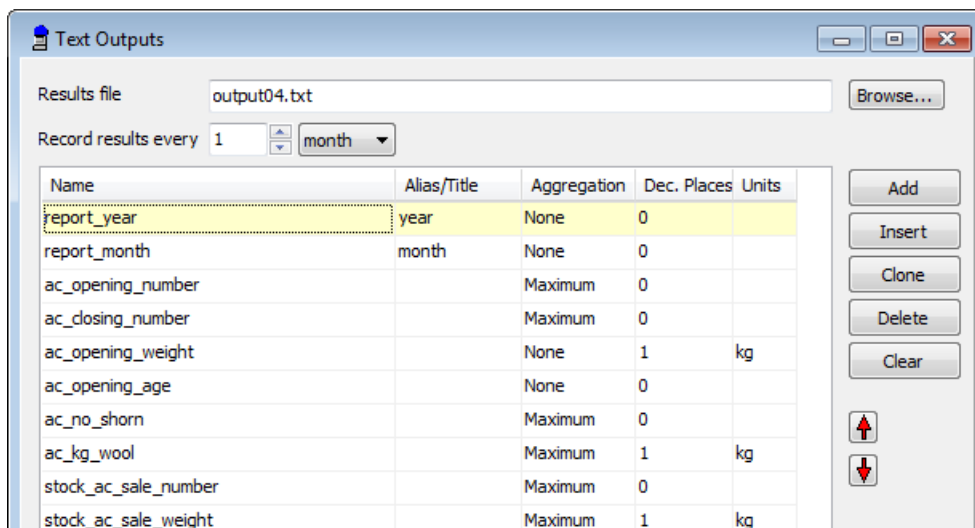
A selected section of script can also be reformatted.



2 Output variables

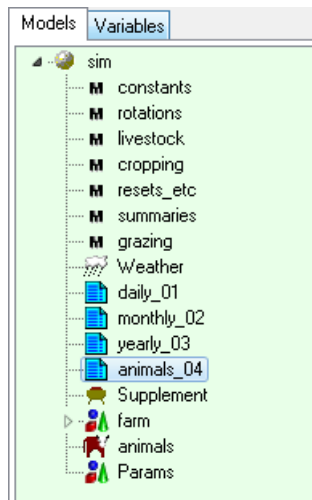
2.1 TextOut component

Output variables listed in the TextOut component now allow units to be set from the TextOut dialog.



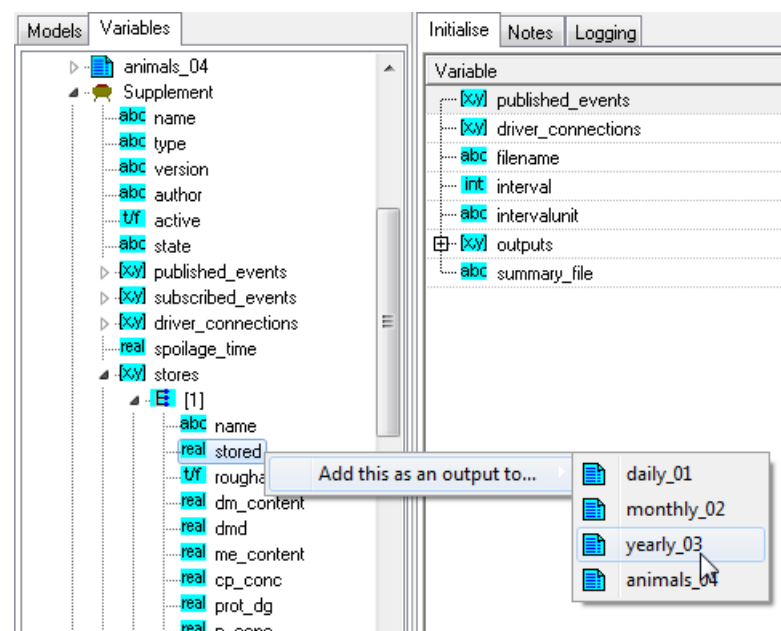
Also included is an option to Clone an existing item in the list.

2.2 Selecting output variables



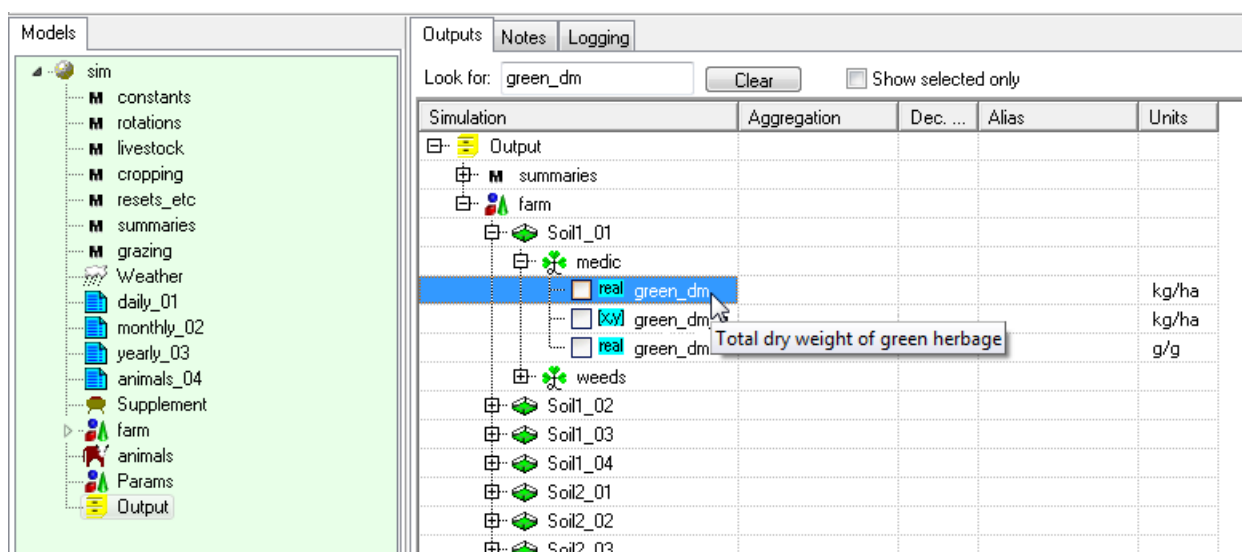
There are two new methods for adding component properties to the outputs list of TextOut components.

1. When a TextOut component is selected, the Variables tab becomes visible. From there the components in the model tree can be expanded and properties can be chosen and added to the output list for any TextOut in the simulation.



Right click on a property and then choose the TextOut component that the variable will added to.

2. Another method of adding variable to TextOut components involves having an Output component (database version) in the model tree.

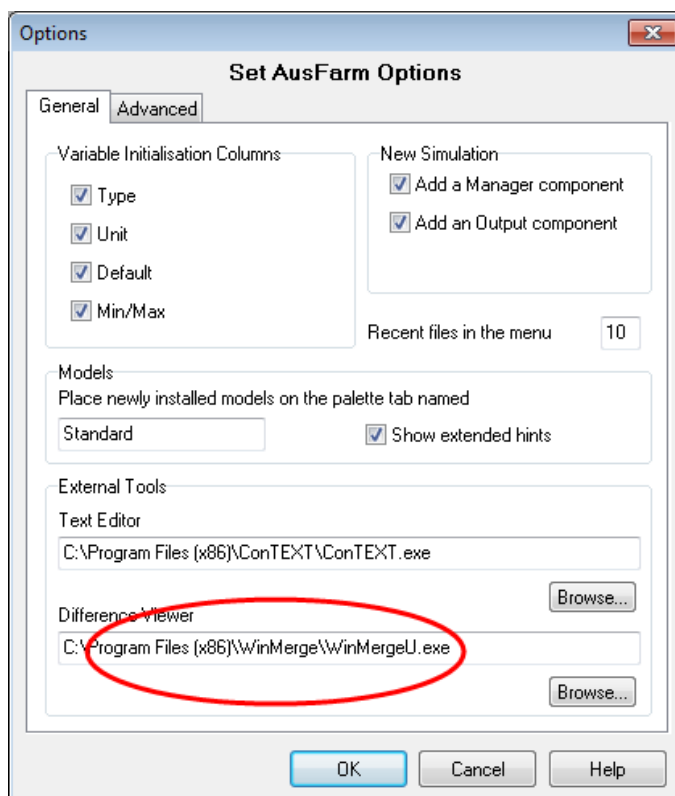


When you select the Output component, the selection tree will become visible in the right-hand panel. From here the variables can be filtered if required. Any variable can be dragged and dropped, using the mouse, onto a TextOut component.

3 Component comparison

It is often useful to compare the initial values for like components in a simulation or between simulations. This can now be done easily from version 1.4.10. Using the clipboard to copy the first component in the model tree and then choosing the Diff menu option on a second component will invoke a difference viewer.

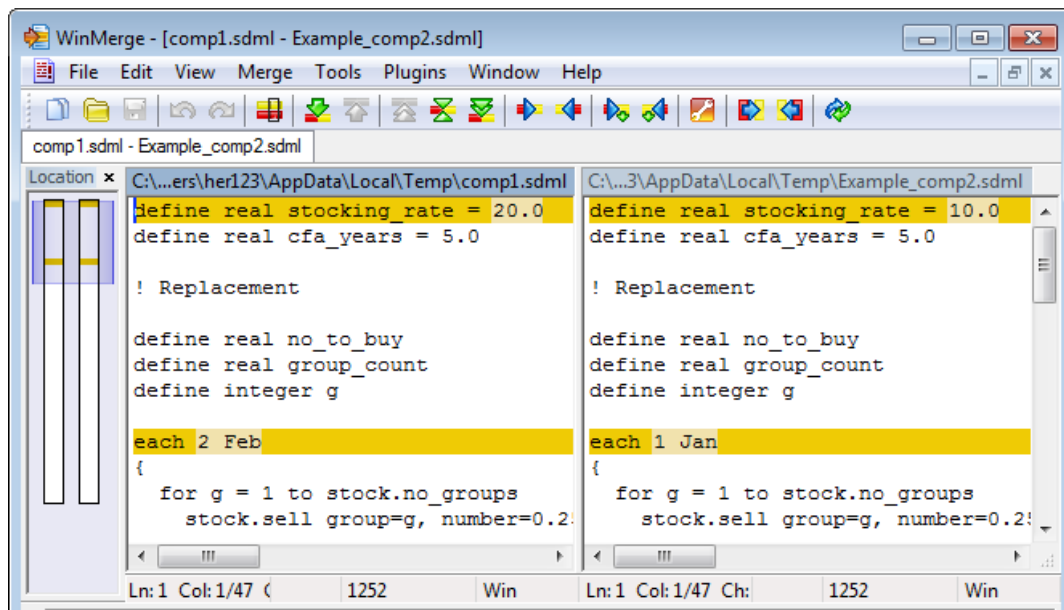
Firstly: Configure the setting on the Options dialog for the Difference viewer.



Then from the model tree select a component and then right click and select the **Copy** option from the popup menu. Select another component in the model tree of the same type and right click the mouse. The popup menu will now have an extra option, **Diff with....**. Choose this option and the external viewer will show the differences between these components. For Manager script components the text will be plain text as seen in the Manager Script editor. For other components the SDML script in XML form will be shown. Although a little cryptic this is still a useful means of checking for differences.

Because this differencing technique uses data stored in the clipboard, it is easy to do comparisons between components in different simulations.

If you change any of the text in one of the difference panels it can be selected as a whole and pasted over the script in AusFarm.



An example of comparing two Manager scripts.

4 Manager scripts

4.1 Multi-line comments

From version 1.4.10, multi-line comments are supported. This means that multiple lines can be commented out using the C-style commenting. In-line comments are also permitted.

Examples:

```

1
2 define integer /*commented text*/ test_int = 0
3
4 /*
5 this is multi line comment
6 that spans more
7 than one
8 line
9 define integer second_int = 0 */
10
11 define integer padd_to_system[7] = [11, /*item 2*/ 12, 13, 21, 22, 23, 24]
```

4.2 New DayOfYear() function

A new inbuilt function has been added to the Manager component. DayOfYear() simplifies script writing by making the setting of day numbers more understandable.

```

8
9 define integer sale_date
10
11 set sale_date = DayOfYear('1-Jan')
12
13
```

Examples:

```

dayofyear('1-Jul')
dayofyear('Dec-31')
```

dayofyear('12 Aug 1961')

Where delimiters can be '-' or ' ' or '/'

Month names must be the first three characters from the English month name.

The Year must be four digits. Where the year is not specified a non-leap year is assumed.

4.3 Constants

4.3.1 Used for initialisation

Constants that are defined in Manager scripts can be used to initialise the size of arrays if necessary. This simplifies the scripts and makes it easier to apply changes to scripts by reducing redundancy of information.

For example:

```
1
2
3 define const integer PADD_COUNT      = 7      ! Actual number of paddocks
4 |
5 define text      padd_name[PADD_COUNT]      = ['farm.hill01', 'farm.hill02', 'farm.hill03', 'fa
6 define integer  padd_to_system[PADD_COUNT]  = [1, 1, 1, 2, 2, 2, 2]
```

Arrays can also have expressions as well as constants for the sizing of the array.

```
14 define integer size = 6
15 define integer flexible[size + 1]
```

4.4 Expressions can be used for initialisation

Variables and constants can be initialised with the results of expressions.

For example:

```
15
16 define text      file_prefix = 'c:\temp\SimpleMixed'
17 define real      file_version = 1.0
18
19 define const string file_name_base = file_prefix & ' v' & str(file_version, '3.1f')
20
21 define const integer join_start = DayOfYear('1-Feb') ! Joining day-of-year for sheep (1 Feb)
22
```

When initialising complex variables it is now possible to use expressions to set values. This can be done within arrays of records.

These are valid now:

```
4 define test = (field1: [sin(20), 901 * 34.56] ; field2: 'Mathematical')
5 |
6 define test2 = (field1: [DayOfYear('1-May'), 222] ; field2: (subfield1: cos(45); subfield2: 3.2 ))
7
```

When initialising an array, each successive element will be assumed to follow the type of the first element. For example:

```

2
3 define test_array = [93.23, 3, 5]      ! floating point values
4
5 define test_array2 = [56, 34.012, 4] ! this is invalid as the second element cannot be stored as an integer

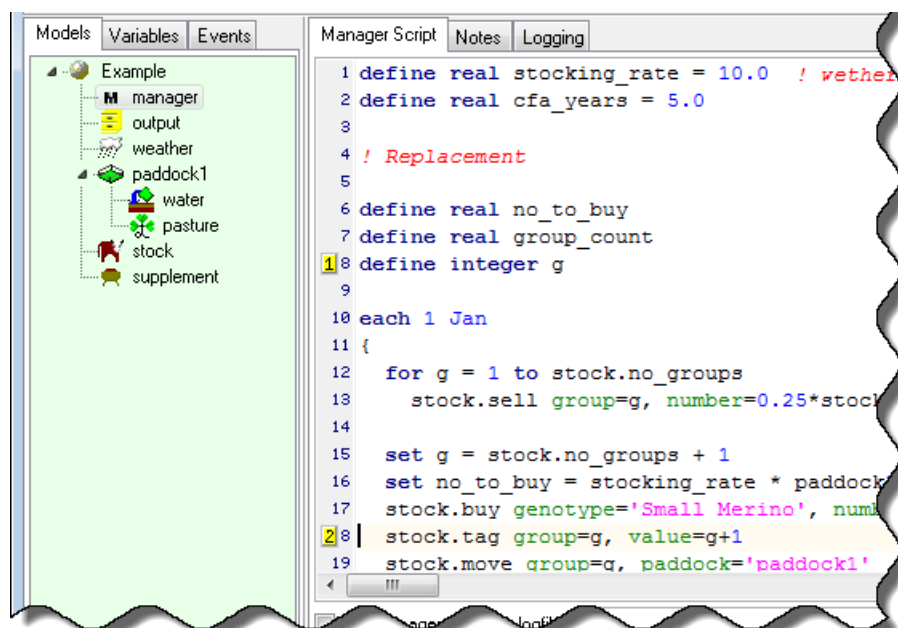
```

4.5 Bookmarks

To set bookmarks in the script there is a keyboard combination that performs this task. To set a bookmark use the key combination, CTRL + Shift + 1. When a bookmark is set you will see a small number icon in the left hand gutter of the editor. To unset the bookmark, ensure the cursor is on the line of the bookmark and use the same key combination. You can have up to nine bookmarks on each Manager editor. Just use the CTRL + Shift + *number* combination for any extra bookmark.

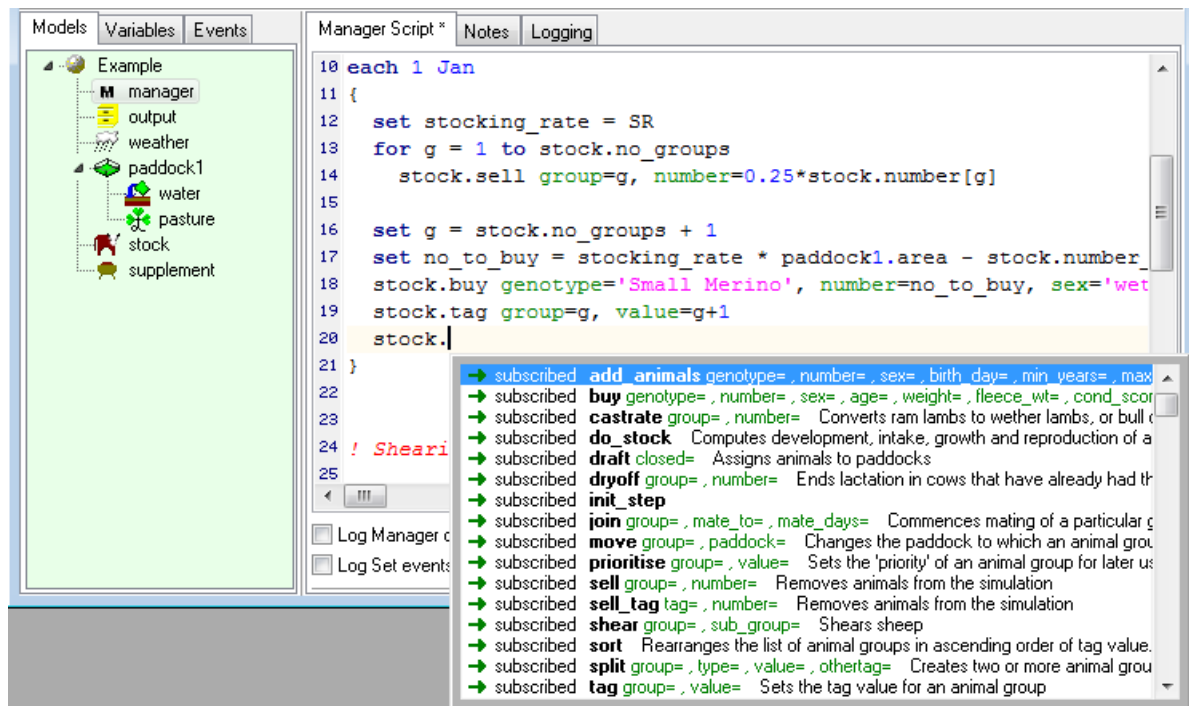
Once a bookmark has been set in a script, it is easy to go to that line at any time using the key combination CTRL + *number*.

Bookmarks are shown in the following figure.



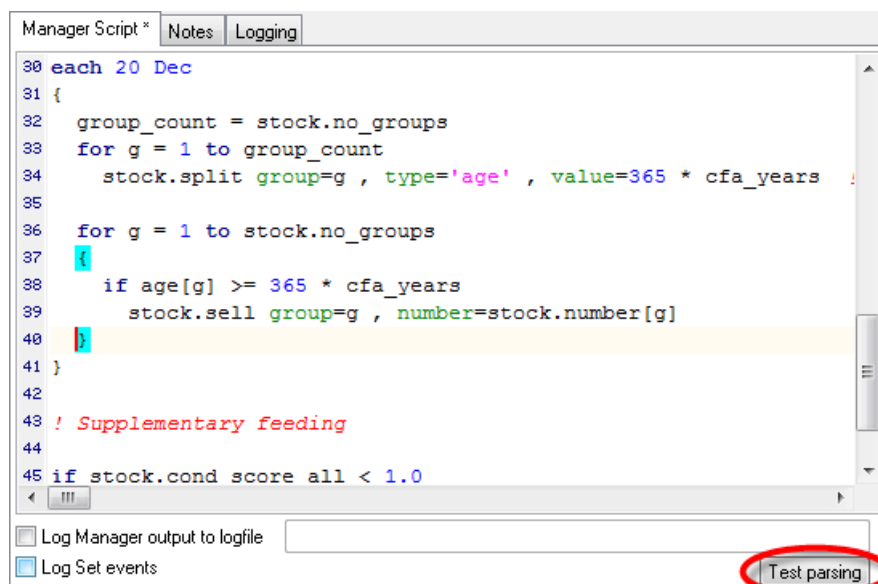
4.6 Code completion

When typing management script and the name of one of the components is followed by a period, by waiting for around one second a window will popup showing a list of properties and events that belong to this component. In the example below you can see a list of the events that can be triggered in the stock component. Highlight the preferred event in the list using the up or down arrow keys on the keyboard or use the mouse cursor to select it. By then pressing enter on the keyboard it will be inserted into the Manager script.



4.7 Matching braces

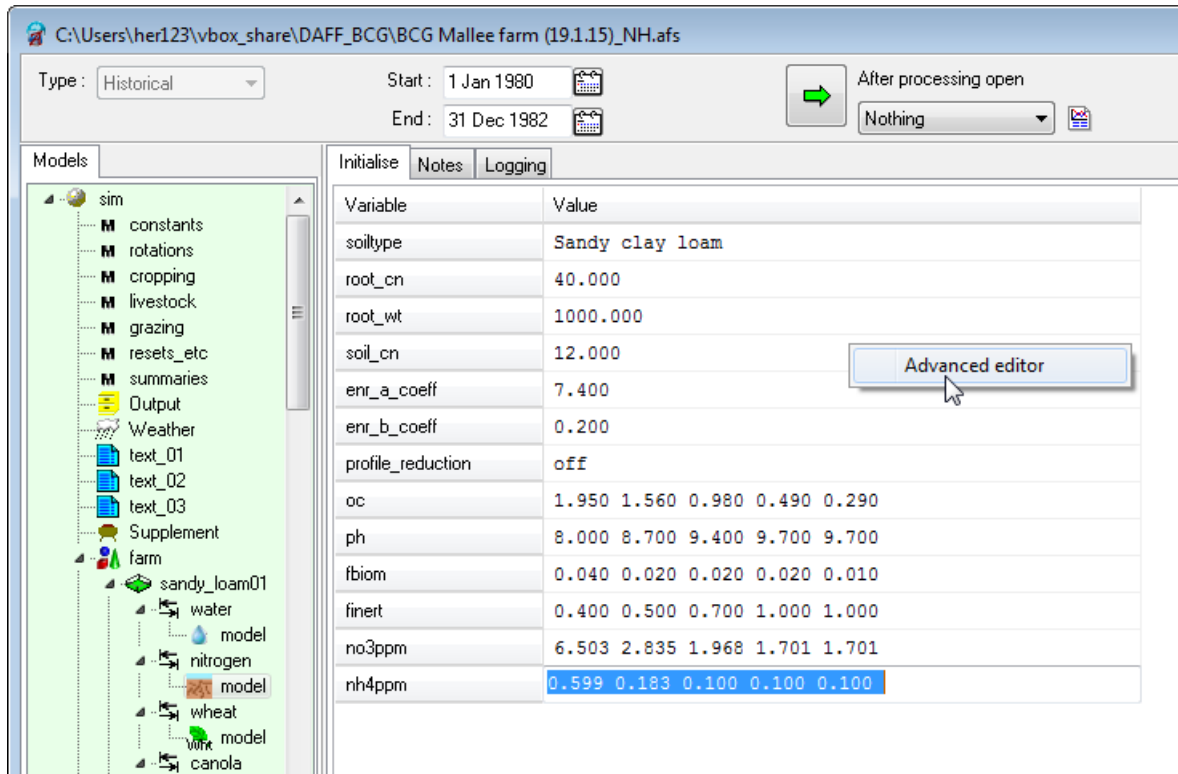
To assist with formatting the Manager script correctly the editor shows clearly the matching braces in the script. When the cursor is placed on a [, {, or (type of braces the corresponding one is also highlighted. As shown in the figure below.



5 Initial property and parameter sections for APSIM components

5.1 Custom editor for APSIM components

AusFarm now includes an improved editor for APSIM initialisation data. By default you will see the values shown in a grid as below. By right clicking the mouse on the grid it is possible to open the advanced XML editor if required. From the XML view it is possible to go back to the grid view by right clicking and choosing **Easy editor**.



The screenshot shows the AusFarm software interface. The top bar displays the file path: C:\Users\her123\vbbox_share\DAFF_BCG\BCG Mallee farm (19.1.15)_NH.afs. Below this, there are fields for 'Type' (set to 'Historical'), 'Start' (1 Jan 1980), 'End' (31 Dec 1982), and a 'Processing' button. A dropdown menu for 'After processing open' is set to 'Nothing'. The left sidebar shows a tree view of models, including 'sim', 'constants', 'rotations', 'cropping', 'livestock', 'grazing', 'resets_etc', 'summaries', 'Output', 'Weather', 'text_01', 'text_02', 'text_03', 'Supplement', 'farm', 'sandy_loam01', 'water', 'model', 'nitrogen', 'model', 'wheat', 'model', 'vink', 'model', and 'canola'. The main area displays a table of variables and their values. A right-click context menu is open over the 'nh4ppm' row, showing the 'Advanced editor' option.

Variable	Value
soiltype	Sandy clay loam
root_cn	40.000
root_wt	1000.000
soil_cn	12.000
enr_a_coeff	7.400
enr_b_coeff	0.200
profile_reduction	off
oc	1.950 1.560 0.980 0.490 0.290
ph	8.000 8.700 9.400 9.700 9.700
fbiom	0.040 0.020 0.020 0.020 0.010
finert	0.400 0.500 0.700 1.000 1.000
no3ppm	6.503 2.835 1.968 1.701 1.701
nh4ppm	0.599 0.183 0.100 0.100 0.100

5.2 APSIM Component Initial Values

APSIM components use xml script to define the initial property settings and model parameters. Previously in AusFarm the initial values and parameters have been visible in xml text form. This has been changed so that only the initial values are now visible. AusFarm will still function with the previous form but the new form is more reliable.

The previous view:

```

001 <initdata>
002   <name>pasture</name>
003   <type>pasture</type>
004   <mass>1000.0</mass>
005   <cnr> 60.0</cnr>
006   <standing_fraction>0.0</standing_fraction>
007
008   <!-- ==== Standard constants below this point ==== -->
009
010   <crit_residue_wt> 2000.0 </crit_residue_wt>
011   <opt_temp> 20.0 </opt_temp>
012   <cum_eos_max> 20.0 </cum_eos_max>
013   <cnrf_coeff> 0.277 </cnrf_coeff>
014   <cnrf_optcn> 25.0 </cnrf_optcn>
015   <c_fract> 0.4 </c_fract>
016   <leach_rain_tot> 25.0 </leach_rain_tot>
017   <min_rain_to_leach> 10.0 </min_rain_to_leach>
018   <crit_min_surfom_orgC units="kg/ha"> 0.004 </crit_min_surfom_orgC>
019   <default_cpr> 0.0 </default_cpr>
020   <default_standing_fraction> 0.0 </default_standing_fraction>
021   <standing_extinct_coeff> 1.0 </standing_extinct_coeff>

```

When you drop an APSIM component into the model tree now this is the XML that you would see from the advanced editor:

```

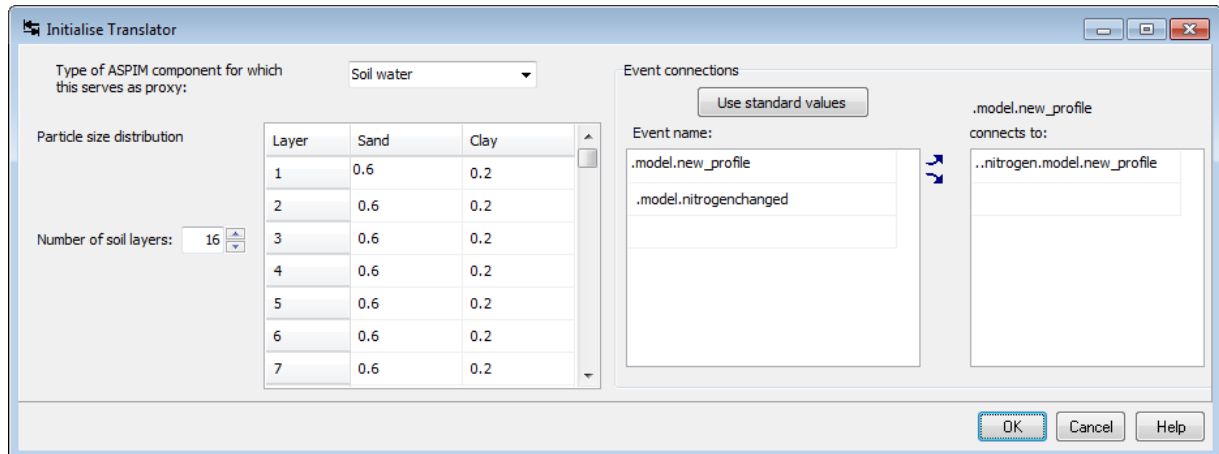
001 <initdata>
002   <name></name>
003   <type></type>
004   <mass></mass>
005   <cnr></cnr>
006   <cpr></cpr>
007   <standing_fraction></standing_fraction>
008
009   <!-- ==== Standard constants below this point ==== -->
010   <!--[model]-->
011 </initdata>

```

The macro `<!-- [model] -->` replaces the parameter section. This will ensure that your APSIM components are always using the correct parameters for the model installed in your tool palette. All that is required is for the initial values to be filled in as shown in the previous image.

6 Translator initialisation dialog

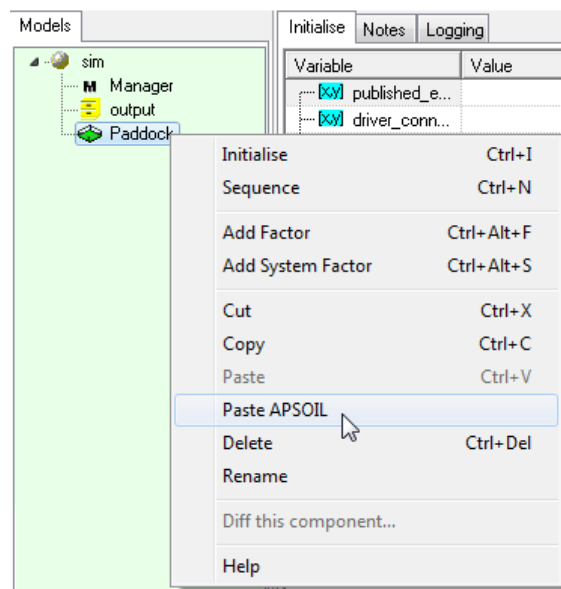
The APSIM component translator component now has an initialisation dialog. This makes it much easier to configure the translator. The dialog is shown below.



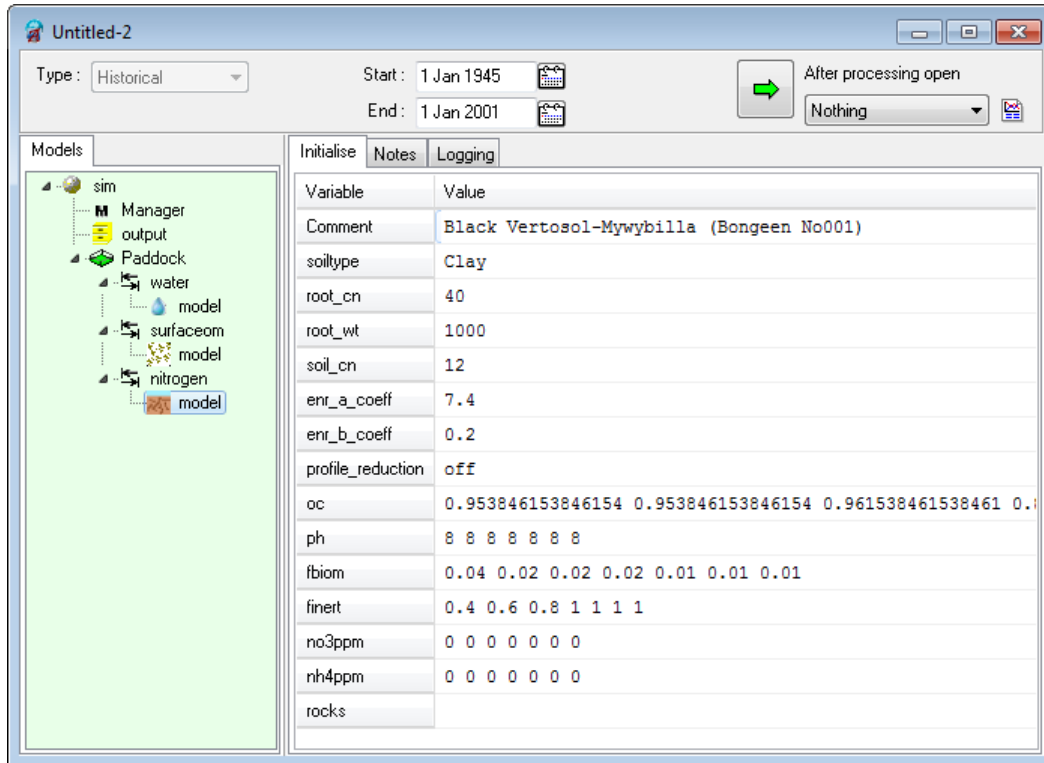
More information in this can be found in the document; *AusFarm User Notes #1 .pdf*

7 Using APSOIL soil data

It is possible to copy the soil descriptions from APSOIL directly into AusFarm. This is done firstly by copying the soil in APSOIL to the clipboard. Then with a paddock component in the model tree in AusFarm; right click on the paddock and choose **Paste APSOIL**.



Once the soil is pasted on the paddock, new components will be added to the paddock system. The nitrogen model then needs to be initialised for initial nitrogen values. The soiln initial values are shown below.



The screenshot shows the 'Initialise' tab of a software interface. The left pane displays a hierarchical tree structure under 'sim', including 'Manager', 'output', 'Paddock', 'water', 'model', 'surfaceom', 'model', 'nitrogen', and 'model'. The right pane shows a table of variables and their values for 'Black Vertosol-Mywybilla (Bongeen No001)'.

Variable	Value
Comment	Black Vertosol-Mywybilla (Bongeen No001)
soiltype	Clay
root_cn	40
root_wt	1000
soil_cn	12
enr_a_coeff	7.4
enr_b_coeff	0.2
profile_reduction	off
oc	0.953846153846154 0.953846153846154 0.961538461538461 0.961538461538461 0.961538461538461 0.961538461538461 0.961538461538461 0.961538461538461
ph	8 8 8 8 8 8 8 8
fbiom	0.04 0.02 0.02 0.02 0.02 0.01 0.01 0.01
finert	0.4 0.6 0.8 1 1 1 1 1
no3ppm	0 0 0 0 0 0 0 0
nh4ppm	0 0 0 0 0 0 0 0
rocks	

The initial values for the cropping modules are also set if they are found in the paddock. LL, KL and XF values for crops will be filled with calculated values if they are not found in the ApSoil description.