# Introduction

EnhancedTemplateProcessor is a tool for tool for creating model input files from a model based on a template. EnhancedTemplateProcessor can replace a formula in the template with the value derived by evaluating that formula. It can also optionally replace parameter names with the values associated with values associated with those names or replace a reference to an array with a value from an array.

# Usage:

EnhancedTemplateProcessor <template name> [<PVAL file name>][<Arrays file name>]

Run EnhancedTemplateProcessor from the command line followed by the name of a template file and (optionally) the names of a PVAL file containing the names and values of the parameters that should be substituted into the file and a Arrays file name containing the names of arrays, their dimensions, and the names of files containing their data. If the file names contain whitespace, the names must be enclosed in quotation marks. If the file names do not contain whitespace, quotation marks around the file names are optional. The template file name must contain an extension. The output of the program is a file with the same name as the template except that the extension will have been removed from the file name. The contents of the output file will be the contents of the input file after having been processed.

# PVAL File format

The PVAL file format is the same as documented for MODFLOW-2005. However, If a line starts with “#--“, the remainder of the line will be treated as defining a parameter for the purposes of EnhancedTemplateProcessor. All such lines must follow the lines defining parameters for MODFLOW-2005 for to ensure that the lines will be ignored by MODFLOW-2005.

# Arrays File format

The first line in the Arrays file must contain a single character that will be used to delineate array substitutions within the template.

The remaining lines in the arrays file that are empty or start with the character “#” will be skipped. All other lines define three-dimensional arrays of real numbers. Each such line has the name of an array immediately followed by an open bracket character “[“. The open bracket character is followed by three positive integers and then a close bracket character “]”. The integers represent the number of layers, rows, and columns in the array. The array dimensions must be separated by commas and/or one or more spaces. Following the close bracket character is the name of a file containing the data for the array. Each line of this file must contain one or more or more values for the array. The values must be separated by commas and/or one or more space characters. When reading the array values, the column index will be incremented most frequently and the layer index will be incremented least frequently. Array names are not case sensitive. Array names may consist only of the characters “A” through” Z”, “a” through “z”, “0” through”9” and “\_”.

# Template file format

In each line of a template file, the first operation to be performed is to replace references to arrays with the array values. Each array reference must be preceded and followed by the array delineator defined on the first line of the array file. The number of spaces between the initial array delineator and the beginning of the array name determines the number of characters used to print the array value. In the text between the array delineators, the array name must appear immediately followed by the an open bracket character, the layer, row, and column indices and the closed bracket character. The corresponding value will be printed, right justified, in place of the text between the array delineators inclusive.

If parameter names are to be replaced by parameter values, the first line of the file must begin with either "ptf " or "jtf " followed by a single character. The character, known as the “parameter delimiter”, is used to specify locations in the file at which parameter names are replaced by parameter values. The parameter names must be surrounded by a pair of the parameter delimiters. Extra spaces area allowed and encouraged before and after the parameter name but within the pair of the parameter delimiters. When the parameter name is replaced, everything between the parameter delimiters and the delimiters themselves will be replaced by the parameter value. If the parameter value is too long to fit within the available space, it will be truncated to fit.

The next line of the template (or the first line, if the parameter delimiter line is not included) must begin with "etf " followed by a single character. This character is the formula delimiter. It plays a role similar to the parameter delimiter. A formula should be included between a pair of formula delimiters. The formula delimiter must be different from the parameter delimiter. However, the width of the available space is indicated differently from how it is done with parameter delimiters. The available space extends from the first formula delimiter through the first character before the beginning of the formula.

The parameter delimiter must be different from the formula delimiter.

The parameter delimiter character can not occur anywhere in the template except where it functions as a parameter delimiter.

The formula delimiter character can not occur anywhere in the template except where it functions as a formula delimiter.

There is no restriction imposed by EnhancedTemplateProcessor on the length of lines in template or PVAL files.

There is no restriction imposed by EnhancedTemplateProcessor on the length of parameter names in PVAL files or template files.

Parameter names must not include whitespace.

# Formulas

Formulas must evaluate to a real number. Note that while logical operations are possible, a formula that depends on parameter values should be a continuous function of the parameters. Failing to follow this rule can result in a failure of the parameter estimation process.

## Operators

The following operators are available in formulas

Table 1. Operators in ModelMuse Formulas

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Meaning** | **Data Types** | **Result type** |
|  | | | |
| = | equals | real numbers, integers, Booleans, text | Boolean |
| <> | not equals | real numbers, integers, Booleans, text | Boolean |
| > | greater than | real numbers, integers, Booleans, text | Boolean |
| < | less than | real numbers, integers, Booleans, text | Boolean |
| >= | greater than or equals | real numbers, integers, Booleans, text | Boolean |
| <= | less than or equals | real numbers, integers, Booleans, text | Boolean |
| and | and | Booleans | Boolean |
| or | or | Booleans | Boolean |
| xor | exclusive or | Booleans | Boolean |
| not | not | Booleans | Boolean |
| mod | modulus (remainder) | integers | integer |
| div | integer division | integers | integer |
| ^ | raise a number to a power | real numbers, integers | real number |
| \*\* | raise a number to a power | real numbers, integers | real number |
| \* | multiplication | real numbers, integers | real number, integer |
| / | division | real numbers, integers | real number, integer |
| + | addition or concatenation | real numbers, integers, text | real number, integer, text |
| - | subtraction | real numbers, integers | real number, integer |

The operator precedence rules are shown in table 2. Operators that are part of the same group have equal precedence. Operators of equal precedence are evaluated in order from left to right.

Table 2. Operator precedence rules

|  |  |
| --- | --- |
| Operators | Precedence |
| ( ) | first (highest) |
| not, ^, \*\* | second |
| and, mod, div, \*, / | third |
| or, xor, +, - | fourth |
| =, <>, >, <, >=, <= | fifth (lowest) |

## Functions

The same logical, math, text, and trigonometric functions available in ModelMuse are also available in EnhancedTemplateProcessor.

# Description of operations

EnhancedTemplateProcessor reads the names of the template file and PVAL file from the command line. (The PVAL file is optional. It then reads the PVAL file, if specified, and associates each parameter name with a real-number value. It then reads the template file and reads the parameter delimiter (if specified) and formula delimiter. It then reads the following lines one by one. In each line, it firsts replaces any parameter names enclosed within parameter delimiters by the associated values. It then reads any formulas in the line, evaluates them, and replaces the formulas with the evaluated value.

EnhancedTemplateProcessor does not have equivalents of the PRECIS or DPOINT variables in PEST.

# Example

## Example Template

ptf @

etf !

this is a line with nothing to replace in it.

This is a line with a parameter value "@ HK1@"

This is a line with a formula "! 2/3\*100000!"

This is a line with formula containing two parameters and a formula "! @ HK2 @ + @ HK3 @!"

## Example PVAL file

# PVAL file created on 8/26/2009 by ModelMuse version 2.0.0.26.

18

HK1 1

HK2 0.01

HK3 0.0001

HK4 1E-6

VKA12\_1 0.25

VKA12\_2 0.0025

VKA12\_3 2.5E-5

VKA12\_4 2.5E-7

VKA3\_1 1

VKA3\_2 0.01

VKA3\_3 0.0001

VKA3\_4 1E-6

KDEP\_Par1 0.9

LVDA\_Par1 1

GHB 1

DRAIN 1

RCH 0.00031

ETM 0.0004

## Example Output file

this is a line with nothing to replace in it.

This is a line with a parameter value " 1"

This is a line with a formula "6.66E4"

This is a line with formula containing two parameters and a formula "0.0101"