

This document was prepared in conjunction with work accomplished under Contract No. DE-AC09-96SR18500 with the U. S. Department of Energy.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This report has been reproduced directly from the best available copy.

Available for sale to the public, in paper, from: U.S. Department of Commerce, National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161,
phone: (800) 553-6847,
fax: (703) 605-6900
email: orders@ntis.fedworld.gov
online ordering: <http://www.ntis.gov/help/index.asp>

Available electronically at <http://www.osti.gov/bridge>
Available for a processing fee to U.S. Department of Energy and its contractors, in paper, from: U.S. Department of Energy, Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831-0062,
phone: (865)576-8401,
fax: (865)576-5728
email: reports@adonis.osti.gov

**Database/Template Protocol to Automate
Development of Complex Environmental Input Models**

by

LEONARD B. COLLARD

Westinghouse Savannah River Company
Savannah River Technology Center
Aiken, SC 29808

A paper proposed for presentation at the
SCS Spring Simulation Multiconference
San Diego, California
April 2 – 8, 2005

and for publication in the proceedings of the meeting

This paper was prepared in connection with work done under Contract No. DE-AC09 96SR18500 with the U. S. Department of Energy. By acceptance of this paper, the publisher and/or recipient acknowledges the U. S. Government's right to retain a nonexclusive, royalty-free license in and to any copyright covering this paper, along with the right to reproduce and to authorize others to reproduce all or part of the copyrighted paper.

Database/Template Protocol to Automate Development of Complex Environmental Input Models

Leonard B. Collard

Westinghouse Savannah River Company
Aiken, SC 29802

Keywords: Contaminant transport, database, automation, batch

ABSTRACT

At the U.S. Department of Energy Savannah River Site, complex environmental models were required to analyze the performance of a suite of radionuclides, including decay chains consisting of multiple radionuclides. To facilitate preparation of the model for each radionuclide a sophisticated protocol was established to link a database containing material information with a template. The protocol consists of data and special commands in the template, control information in the database and key selection information in the database. A preprocessor program reads a template, incorporates the appropriate information from the database and generates the final model.

In effect, the database/template protocol forms a command language. That command language typically allows the user to perform multiple independent analyses merely by setting environmental variables to identify the nuclides to be analyzed and having the template reference those environmental variables. The environmental variables can be set by a batch or script that serves as a shell to analyze each radionuclide in a separate subdirectory (if desired) and to conduct any preprocessing and postprocessing functions.

The user has complete control to generate the database and how it interacts with the template. This protocol was valuable for analyzing multiple radionuclides for a single disposal unit. It can easily be applied for other disposal units, to uncertainty studies, and to sensitivity studies. The protocol can be applied to any type of model input for any computer program. A primary advantage of this protocol is that it does not require any programming or compiling while providing robust applicability.

INTRODUCTION

A commercially available program, Porflow™ (ACRI, 2004), was used to analyze the transport of each radionuclide from a trench containing solid waste to a hypothetical well 100 m away (Collard and Hiergesell,

2004). Porflow™ required an input file, designated as the model in this document.

A preprocessor program was developed to read a template, incorporate the appropriate information from a database and generate the final model. Special commands in the template contain information that direct the preprocessor perform matching and copying actions. Controls in the database specify the variables in the command to match with specified variables in the database. Other database controls specify the variables in the command and database to write and the order to write them. The database also contains controls that specify a default format for outputting the selected information. The template can override the database controls.

A preprocessor (prePorflow, a Fortran computer program) uses the protocol to generate a model for each radionuclide. The primary intent of the protocol was to produce a simple but sophisticated tool that

- eliminates computer programming (except for the preprocessor)
- allows a single set of databases for multiple radionuclides and multiple disposal units
- provides options for matching template data with database keys
- provides options to select template and database data to output to the model
- provides options to select the output format for the model

The system using the protocol is depicted in Figure 1. Each component of the system utilizing the protocol is described below.

SHELL

The dashed box represents the iterative portion of the system referred to as the shell. Multiple radionuclides were individually analyzed, so a loop was established to select each radionuclide in turn. For the initial application, only one disposal unit was analyzed. If more disposal units are considered a second outer loop representing each disposal unit will be required. After selecting the next radionuclide from a list, environmental variables were set describing that radionuclide. A separate directory was created to hold all information for the analysis of the selected radionuclide.

The default directory was changed to the newly created directory by moving to the newly created directory. The iteration was performed by recursively calling a DOS batch

file on an IBM PC using the pseudo-code segment shown in Figure 2. A script file in Unix or similar shell programming can accomplish the same purpose.

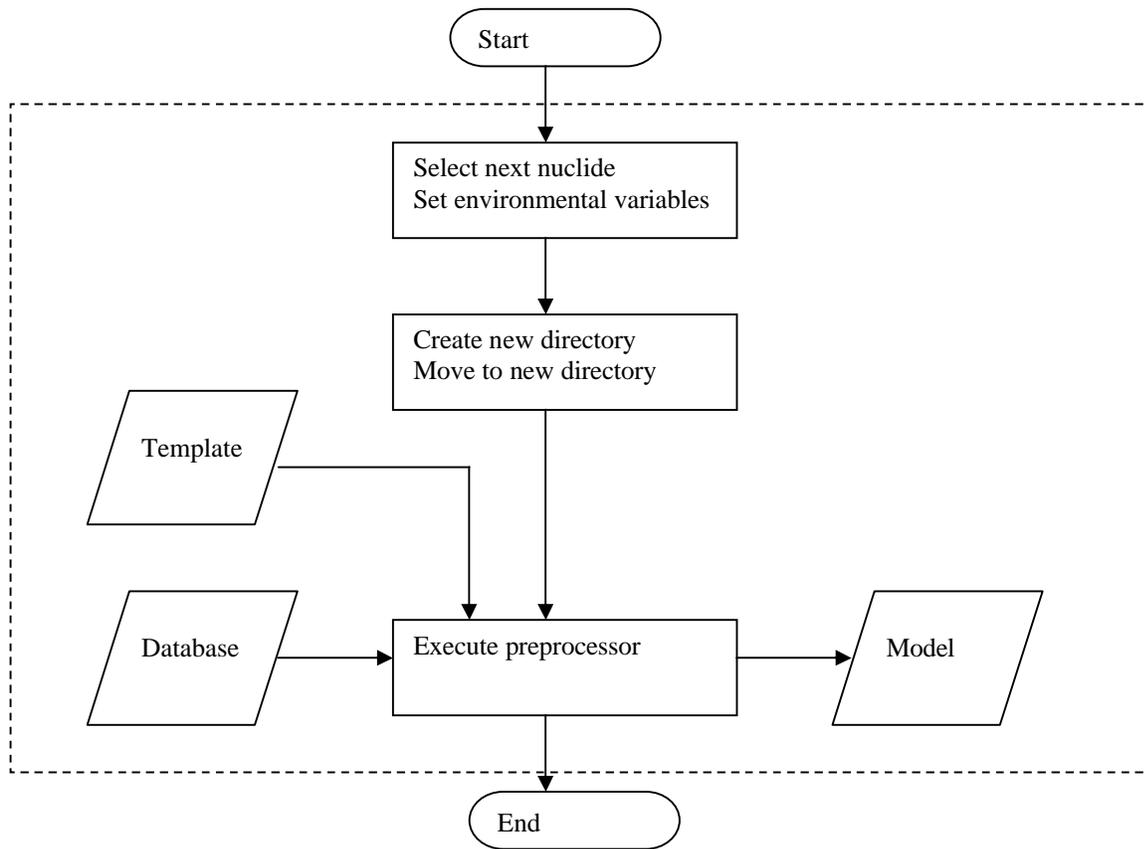


Figure 1. Execution flowchart

```

:: ***** recursive call
%1 %2
:: ***** Loop for each nuclide
For %%N in (Nuc1 Nuc2) DO call %0 Goto NextNuc %%N
Goto End

:: ***** Start of operations for new nuclide
:NextNuc
:: ***** set environmental variable identifying new nuclide
set Nuclide=%3

:: ***** Make new directory and change to it
md %Nuclide%
cd %Nuclide%

:: ***** Execute preprocessor
prePorflow.exe
...
:End
    
```

Figure 2. Batch code segment for recursive calls

PREPROCESSOR

The preprocessor reads records from the template. If the record does not contain a special command to activate the database linkage, then that record is merely echoed to the model. If the template record contains a special command, the preprocessor performs one of several special functions. The major function is to interact with the database and select replacement values to insert into the model, similar to a mail-merge program, but with greater capabilities. The preprocessor can perform auxiliary functions to enhance the database selection and insertion capabilities.

TEMPLATE

The template protocol includes not only the command itself, but also the strings that follow on the same special command line or its continuation. Table 1 provides a summary of the most important special commands with a

brief description of the actions invoked by that command.

Each special command is discussed in more detail below.

Table 1. Special template commands and actions

Category	Special command	Action
Error checking	!BuildEcho	Echoes special commands to the model.
	!BuildNoEcho	Turns off echo of special commands to the model.
Variable definition	!Build~Set	Declares a variable and sets a value for it. The variable can be used later where its current value is placed in the model. This improves the readability of the template and provides flexibility.
	!Build~Sub	Redefines the value for a previously declared variable.
Logic control	!BuildIf	The capability to have if-else-endIf statements is provided. Combining this capability with variable definitions increases flexibility.
	!BuildElse	
	!BuildEndIf	
Database	!Build	Perform substitution from database
	!BuildChain	Perform substitution from database for a chain
	!BuildChainContinue	Continuation of !BuildChain command
	!BuildChainEnd	End of !BuildChain command

Error checking commands

The first two special commands allow specific error checking of the template. By default the preprocessor, prePorflow, echoes all special commands to the model. Porflow™ ignores all input that begins with an exclamation point, thus inclusion of the strings containing the special commands in the model does not affect the actual analysis of the model. Turning off the echo produces a model that does not contain the special commands, equivalent to not using the preprocessor. The echo can be turned on for a specific portion of the template which allows for specific error checking.

Variable definition commands

The third and fourth special commands allow variable declaration, definition and redefinition. The template subsequently can refer to these variables, which will be replaced by their contents. These capabilities provide a simple mini-programming capability that does not require compilation of a program. The !Build~Set special command creates a variable and defines its initial value. Because the protocol is intended for an individual without programming capabilities, the number of variables is limited

to 10 and a maximum length of 30 characters is allowed for each variable name (which is stored as a string) and the name must begin with a tilde (~). The value associated with each variable is also a string with a maximum length of 30 characters allowed. These limitations could easily be relaxed, but they worked very effectively for the analysis where they were applied. The !Build~Sub special command merely redefines the value for a previously declared variable. All strings are enclosed by double quotes.

Examples of the use of the first special command are

```
!Build~Set    "~Nuclide1"    "Pu-241"
!Build~Set    "~Nuclide2"    "Am-241"
!Build~Set    "~Nuclide3"    "Np-237"
```

This set of commands could be used to define a decay chain. The template could be generic and use "~Nuclide1", "~Nuclide2", and "~Nuclide3" throughout the remainder of the template, thus making it applicable for all chains and only requiring the initial variable definitions.

Logic control commands

The template protocol allows use of an if-else-endIf capability. Note that the looping capability to analyze each

radionuclide must be provided via the shell. The shell already needs to provide a looping capability to be able to create and change directories, so no additional looping capability was provided within the template protocol. The !BuildIf special command compares two strings. The only option is to specify whether the test is to see if the strings match (using .EQ.) or if they do not match (using .NE.). If the test condition is true, then that portion of the template is operated on. If the condition is false, then the portion of the template after the !BuildElse is operated on (if present). A simple example of an if-else-endif logic control follows:

```
!BuildIf      "~Nuclide2"      "EQ."  ""
!Build~Set   "NumberOfNuclides" "1"
!BuildElse
!Build~Set   "NumberOfNuclides" "2"
!BuildEndIf
```

The example determines the number of nuclides in the chain. If the name of the second nuclide is empty, then the number of nuclides is set to one, else it is set to two. This example works correctly only for decay chains with two or fewer radionuclides.

Database commands

The database category of special commands in the template forms the core of the protocol. These special commands interact with the database to produce the main information for the model. The first special command is !Build. This special command will be followed by several strings. Any string that begins with a tilde (~) is replaced by its value, because that string should have previously been defined. The first string after the !Build special command specifies the name of the database to use for matching. The database name is appended to a pathname that the preprocessor reads only once. This concatenation of names allows shorter names to be used in the template and requires the pathname to be entered only once.

Subsequent strings on the special command line can be used for matching with database data, copying to the model or merely as comments. The same string can be used for matching and for copying. A simple example for a !Build command is as follows:

```
! DB:2str 2num/ Match:2 DB=1,2 In=2,3/ OUT: In:2 @2,3; DB:0str; 2num @1,2
      2 2
      2 1,2 2,3
      2 2,3 0 2 1,2
( 'TRAN for ', a,1x,a, ' C Kd= ', es9.2, ' diff= ',es10.3 )
"H-3      " "Topsoil      " 0.00E+00      1.578E+02
"H-3      " "Clay         " 0.00E+00      4.734E+01
```

```
!Build "Kd.dat" "I-129" "Sand" "Use for waste also"
```

In the example, the database containing Kd values would be used for matching. The radionuclide used for matching would be "I-129". the physical medium would be "Sand", and "Use for waste also" would be a comment. The order of the strings depends on the matching order specified in the database.

!BuildChain provides a method to copy the set of all decay information for a decay chain. The special command contains the name of the parent radionuclide. The preprocessor then uses the set of variable names for the chain. For each radionuclide in the chain, the half life is read and copied to the model. Porflow™ requires regeneration information for the progeny. The regeneration is unity if no branching fraction is involved and the analysis is performed using moles or atoms. For analyses using Ci, the ratio of parent to daughter half lives is required and can easily be provided in the database for inclusion in the model.

!BuildChainContinue is provided to continue information across more than one input line, because the commands can become quite lengthy. The !BuildChainEnd command signifies the end of the !BuildChain.

DATABASE

The database consists of a separate file for each type of material property, e.g., half life, radionuclides in the decay chains, distribution coefficients describing the partitioning of a radionuclide between soil and water, etc. Each database contains three or four sections of information as follows:

1. header section to describe template and database file layouts, and provide matching controls
2. format section to describe output to model
3. data for matching, output and comments
4. end comments

A partial database example is provided in Figure 3.

Figure 3 Database example

The example header section is the first four lines of the database. The first line is strictly a comment line to help improve readability. A brief description is provided for each number in the rest of the header section. The numbers in the first record are vertically aligned with the subsequent numbers to easily see the relationship, although this is not a requirement.

The second line describes the database layout. For this case the database contains two strings followed by two real numbers. This layout is seen for the first data line that in this example starts with "H-3". The first string is the radionuclide, the second string is the physical medium (Topsoil). The two real numbers are the distribution coefficient (Kd) and the diffusion coefficient.

The third line of the database describes the matching protocol. The first number indicates that 2 fields need to match. The next pair of numbers defines the positions in the database that are used for matching, in this case the first and second strings are selected. The second pair of numbers defines the positions in the template that are used for matching. In this case the second and third strings in the template are used for matching. The first string in the template was used to provide the name of the database, so the next strings after that are used for matching. Matching is performed in the order in which the strings are presented.

The fourth line of the database describes the order in which to output information to the model. The first set of values describes information to copy from the template (the "In" in the first record refers to the template as input). In this example two values will be copied from the template. The positions are the second and third strings. The second set of values describes information to copy from the database itself (the "DB" in the first record refers to the database). The next number describes how many strings from the database will be copied. Because the value is zero, no positions in the database are entered. After the zero, the next number of 2 describes how many real numbers from the database will be copied. The 1,2 that follows states that

both numbers from the database will be copied in the order entered.

Because detailed information is provided for each match and each output, information for the database or the template can be used both for matching and output, and the order can be changed. For example, the matching could be done for the second string, then the third string, but they could be output in the opposite order.

The format section follows the header section. The format is any valid Fortran format. In this case the literal 'TRAN for ' is output. Then two strings are output, which are separated by a space. Referring to the last line of the header section, the two strings are the second and third strings from the template. The literal ' C Kd= ' is next output followed by a real number in scientific notation. Finally the literal ' diff= ' is output followed by a second real number in scientific notation. Referring to the last line of the header section, the two real numbers are the first and second numbers in the database, respectively.

The data section for matching, output and comments follows the format section. The first line of data was described in the discussion of the header section.

The end comments section is the last section, which is optional. If a \$EndData line appears in the database, the preprocessor stops processing, so the database can contain extensive comments on how the protocol works for that database.

PREPROCESSOR – PART 2

Two more features of the preprocessor provide more power. The first feature is that the preprocessor will check the information being copied from the database to see if it consists of special commands, and if so, operates on them before copying them to the model. Figure 4 shows an example for decay chains:

```

! DB:6str 0num/ Match:1 DB=1 In=2/ OUT: In:0; DB:6str @1,2,3,4,5,6 0num
 6 0
      1 1 2
      0 6 1,2,3,4,5,6 0

('Build~Set "~Waste-Spec1" "",a,""/, 'Build~Set "~Generic1" "",a,""/, 'Build~Set "~Elem1" "",a,""/, &
'Build~Set "~Waste-Spec2" "",a,""/, 'Build~Set "~Generic2" "",a,""/, 'Build~Set "~Elem2" "",a,""/)

"U-234_Glass " "U-234 " "U " "Th-230" "Th-230" "Th"

```

Figure 4. Database example with special commands

The format section (!Build~Set “...”) indicates that multiple lines of !Build~Set special commands form the output. Because the preprocessor operates on them before performing a copy, no copy actually occurs. Instead, the commands declare and define variables “~Waste-Spec1”, etc. that can be used subsequently by the preprocessor or that can appear subsequently in the template. Only 1 match is performed, so the user generates the decay chain merely by identifying the parent in the template.

The second feature is that the user can override the default protocol in the database. If the override option is selected by including an “Override” string at the end of the !Build special command line, then the entire header section and format section are changed by including them in the template. This feature provides full control to change the operations of the database and how they interact with the template.

MODEL

The model is merely the output from the preprocessor. The model can range from a standard file to a file that includes all the special commands to allow complete error checking.

RESULTS

The preliminary version of the database/template protocol was developed and successfully implemented (Collard and Hiergesell, 2004). The initial implementation indicated the need to provide additional capabilities. Some of those capabilities were incorporated by revising the preprocessor.

CONCLUSIONS

The protocol establishes a balance between simplicity and robustness that satisfied the primary intent to produce a simple but sophisticated tool that

- eliminates computer programming (except for the preprocessor)
- allows a single set of databases for multiple radionuclides and multiple disposal units
- provides options for matching template data with database keys
- provides options to select template and database data to output to the model
- provides options to select the output format for the model.

A simple mail-merge program could not have satisfied the complexities involved. A complex computer program or a

set of simpler computer programs could have produced the same models. However, changes to those programs would have been required for each type of disposal unit analyzed. The protocol described in this document contains the robustness to satisfy the projected needs. With only one database better control of the quality of the information will exist.

While this protocol was developed to satisfy a specific need, it has sufficient robustness that it provides a widespread field for applications without the need for additional programming. This protocol can be applied to uncertainty and sensitivity analyses and is general enough that it can be applied to almost any analysis program.

A set of comprehensive examples is required to educate users on nuances and more sophisticated features. For example, the capability to have a database contain special commands that the preprocessor operates on before copying information to the model requires more thought than does direct copying. The Override capability and sophisticated formatting are more complicated.

A commercial database may be needed to track changes to the data in a more rigorous manner. In that case, the commercial database could produce the preprocessor database as a standard report.

REFERENCES

ACRI, 2004. <http://www.acri-us.com/software/porflow/default.htm>

Collard, L.B. and R.A. Hiergesell, 2004. *Special Analysis: 2004 General Revision of Slit and Engineered Trench Limits*, WSRC-TR-2004-00300, Revision 0, Westinghouse Savannah River Company, Aiken, South Carolina, 29808, June 14.