Debugging TRACE Input Errors

OBJECTIVES

Become familiar with and practice basic input debugging methods. The next two sections provide information useful for TRACE input debugging. Exercises follow these sections. The input model that will be used in this exercise is the simple faucet model described in the presentation discussion. This model adjusts the "hot water" VALVE to get a desired temperature out of the faucet.

TRACE OUTPUT FILES

TRACE writes useful information to some ASCII files when processing an input file. These files are the sources of information about where an error occurred. When the TRACE option '-p <PREFIX>' is used (where <PREFIX> is the base file name used writing output files), the following output files are relevant to finding input errors (SNAP automatically passes the job name to TRACE as the -p option):

- <PREFIX>.out – This file is a verbose TRACE output file that shows component information while processing the input file, as well as any error or warning information. During a simulation, it also shows component state at intervals defined by the 'large edit' time steps among other things.
- <PREFIX>.msg – This file shows warning error and informational messages printed from TRACE. In essence, it is a filtered version of what is printed to the <PREFIX>.out file.
- <PREFIX>.inp – This is the TRACE input file for the current simulation.
- <PREFIX>.echo – This is a filtered version of the <PREFIX>.inp, that highlights the types of values expected in the input. This file sometimes identifies unexpected input values.

When running from SNAP, a <PREFIX>.screen file is also generated that contains

information printed to the 'Job Status' window. The information printed to this file is similar to the 'msg' file. These files are located together in the folder where the TRACE simulation was executed.

Suggestions for Finding Input Errors

The SNAP environment makes it much easier to build or modify a TRACE model with few input errors than it is when doing the process manually. In a small set of cases manual editing may be preferable, but in general it is recommended that SNAP be used to generate and edit TRACE models.

Suggestion 1: Examine all Error and Warning messages reported by TRACE

Sometimes, but not always, there is information surrounding the final 'fatal error' flag that indicates where the 'fatal error' occurred. Typically the relevant information is associated with some previous error or warning message that may point to the cause of the error. Look at each of the error messages to determine if any were the cause of the run failure.

Sometimes an error or warning will indicate a **card number** where the error occurred. This refers to a line number of the input file. Additional information may be printed indicating why TRACE did not like the value. Once the line number where the error occurred is determined, you can proceed to Suggestion 5 below

Often relevant information about an error or warning is displayed before or after the actual error or warning message, so it is a good idea to look carefully at the information surrounding an error or warning.

Errors are reported in the order that they are found in the input file. It is recommended that you start from the first message that is output and work your way down through the errors and warnings, since fatal errors reported at the end may come as a consequence of input errors which occurred earlier in the input file.

Suggestion 2: Look for an Error Identifier in the 'echo' File

The echo file is a copy of the input file with comment cards stripped out. At the end of each line is printed the associated line number in the original input file. The echo file prints the message '*error' on lines where invalid input is identified. If a fatal error occurs on input processing, search for this string in the echo file, and carefully review the TRACE input on the line reported to be in error. Note that the actual error may occur on a line slightly before the one identified as the line with an error.

Suggestion 3: Look at where the Output File Stopped

As an input deck is processed, component information related to the current component being processed is printed to the <PREFIX>.out file. Input file processing sometimes stops when a 'fatal error' occurs. The component where the <PREFIX>.out file stops processing sometimes indicates the source of an input error. If the error and warning messages don't identify the source of the problem, check where processing stopped in the output file, and examine the component input carefully.

Suggestion 4: Compare the Model Against a Previous Working Model

When you have a recent model that ran successfully, and model changes cause the simulation to fail, comparison of the failed input file with a recent working input file can help in identifying the problem.

There are several good text diffing tools available that can be used to compare TRACE ASCII input files. An open source (freely available) option is Winmerge (http://www.winmerge.org). SNAP also has some built-in diffing capability that can be used to compare the content of med files, but not everything in the model is included in the comparison.

A source control program allows you to save old copies of files (such as a TRACE model) as you continue to develop the model. Use of a source control, such as bazaar or subversion, and frequent commits of your model to the source control, is recommended when developing models. The source control allows you to restore an old copy, or retrieve an old copy for comparison. This can be very helpful when problems arise in your model. TortoiseBZR and TortoiseSVN are freely available source control programs that integrate nicely with windows explorer.

Suggestion 5: Once the Component with an Error is Identified, Examine the Component Input

Once a component or section of the input has been identified to contain the error, compare the input against the TRACE manual input for the component to identify invalid values. SNAP can also be useful for examining values since it includes built-in help strings for input values, and the SNAP help system, when set up properly, is linked to the TRACE documentation pdf files.

Input Debugging Exercise Setup

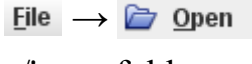
The "Exercise Key" included in the workbook may be useful to help locate the various parts of the SNAP Model Editor that are referred to in this exercise.

Open and submit the faucet model (with errors) by doing the following:

1. Double click on the 'FaucetWithErrors.med' file from the 'Day4/Morning/PWR3_Input_Debugging' folder to open it.
2. Run the faucet model by doing the following:
   a) Locate and click on the "Job Stream" tab at the bottom of the View Window. A Job Stream has already been set up to run this exercise.
   b) Lock the View Window by clicking on the padlock icon  located in the left-hand side of the Toolbar. When locked, the padlock is closed.
   c) Click on the "Execute_Faucet" button  in the View Window to submit the job. A stream warning dialog will appear. For now, ignore the warning and click the "Yes" button. Click the OK button in the Submit Stream dialog box.
   d) The Job Status window should open and display the TRACE screen output once the job is submitted.
   e) After the simulation has run, unlock the View Window by clicking on the padlock.

INPUT DEBUGGING – DIFFING EXERCISE

The simulation should stop with a 'fatal error' message. Note that there is a second model in the same folder with the 'FaucetWithError.med' labeled 'FaucetWorking.med'. The models are nearly the same but the 'FaucetWorking.med' model doesn't fail. We will compare the models to help identify where there might be problems. Typically diffing with a previous working model would be done if you are unable to identify the input error after examining the error and warning messages, but since we will be fixing the errors as we examine them in the steps that follow, we will compare the models first. To compare the models, do the following:

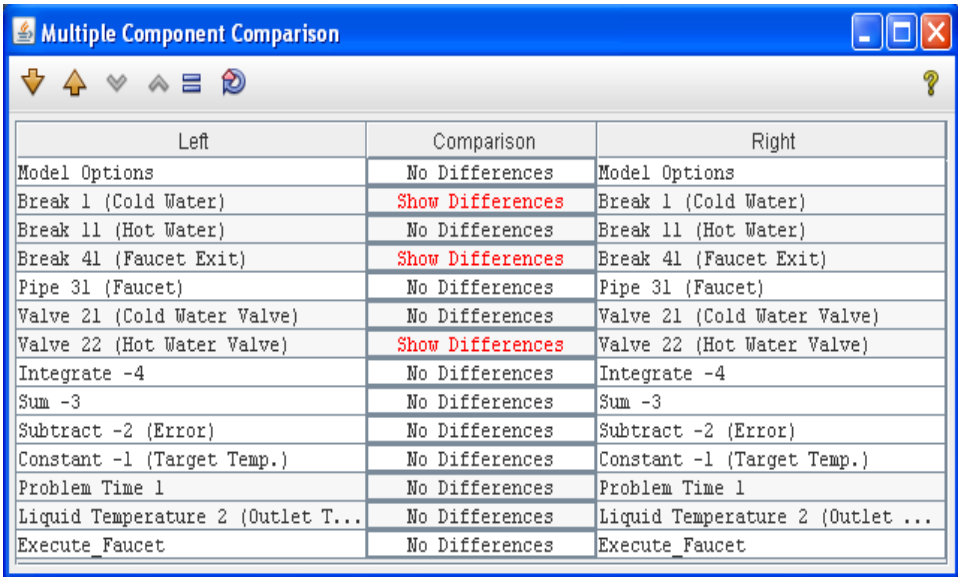
1. In the SNAP Model Editor, select  → , and navigate to the 'Day4/Morning/PWR3_Input_Debugging/' folder, and open the

'FaucetWorking.med' model.

2. Right click on the FaucetWorking.med - (Faucet) title bar, and choose Select Left Side to Compare.

3. Click on the FaucetWithErrors.med - (Faucet) title bar to activate the faucet with errors model.

4. Now right click on the title bar FaucetWithErrors.med - (Faucet) and select Compare to Faucet. The following dialog should appear:



Left	Comparison	Right
Model Options	No Differences	Model Options
Break 1 (Cold Water)	Show Differences	Break 1 (Cold Water)
Break 11 (Hot Water)	No Differences	Break 11 (Hot Water)
Break 41 (Faucet Exit)	Show Differences	Break 41 (Faucet Exit)
Pipe 31 (Faucet)	No Differences	Pipe 31 (Faucet)
Valve 21 (Cold Water Valve)	No Differences	Valve 21 (Cold Water Valve)
Valve 22 (Hot Water Valve)	Show Differences	Valve 22 (Hot Water Valve)
Integrate -4	No Differences	Integrate -4
Sum -3	No Differences	Sum -3
Subtract -2 (Error)	No Differences	Subtract -2 (Error)
Constant -1 (Target Temp.)	No Differences	Constant -1 (Target Temp.)
Problem Time 1	No Differences	Problem Time 1
Liquid Temperature 2 (Outlet T...	No Differences	Liquid Temperature 2 (Outlet ...
Execute_Faucet	No Differences	Execute_Faucet

5. Click on to filter the list such that only components that are different in the two models are listed. This shows that Break 1, Break 41 and Valve 22 are different in the two models.

6. To see the differences in each of the models, click on the associated Show Differences button. Explore the differences to see if this gives you some insight into why the model is failing. However don't make any changes to the model at this point since we want to review the errors output by TRACE and then fix the errors.

The SNAP diff dialog compares the ASCII input for each component in the different models. In the SNAP diff dialog, lines that are different are highlighted in light red. Actual differences in the line are highlighted with red characters. Note that you can also use the SNAP differ to compare components **within** a model as well although instructions are not given herein.

In the ASCII input for a component, SNAP typically prints the TRACE input variable either above an input value, or for values that come in a list, to the far left between asterisks (asterisks are used to bound comments in the TRACE input file). The TRACE input variable is the short identifier used in the TRACE input manual when describing TRACE inputs. For example, 'NCELLS' is the input variable for the number of cells in various components including the Pipe component. SNAP outputs these in lower case.

Note that because there are only a few differences between the working faucet model and the failed faucet model, it is easy to isolate the area where errors might be located, by diffing the models. For this reason, it is suggested that you use a source control while developing models, and commit changes to the source control frequently, particularly after you have a running model. If you do this, it will be much easier to identify when an input error was introduced by finding the latest working TRACE model, and reviewing the changes to the model just following this. This limits the number of changes that have to be reviewed.

7. After you have finished comparing differences close the diffing dialogs, right click on the 

title bar, and select Close to close the working faucet model since we will only be working with the model with errors from this point on.

Input Debugging – Errors and Warnings Exercise

Now review the errors and warnings to try and identify why the model failed. SNAP provides a convenient way of navigating through errors and warnings reported by TRACE. When a fatal error occurs, to explore the errors and warnings do the

following:

1. In the 'Job Status' window that opens when you run the simulation, click on the icon on the toolbar for the simulation. This lists the text files output by TRACE for that simulation.

2. Click on `trcmsg - Faucet.msg`, and the message file should open in a text viewer.

> The '*.msg' file is often preferred for looking up errors in a standard text editor since it is more concise.

3. In the Message Viewer, note the error message:

```
#################
## Input Error ##
#################

*PreInp* input error detected in Faucet.inp on line number 284 column number 11
Cannot parse input value: 7=35
```

   The error occurred on line 284 and input processing could not parse the input value: 7=35.

4. Close the Message Viewer and open the input file `tracin - Faucet.inp` in the same manner as done with the message file (Items 1 and 2 above).

5. Search for the string '7=35' using the “Find” dialog box

6. Remember that you should typically examine information printed above and below the error message to look for clues as to why the input error occurred.

7. We see that the error occurred in a comment line and that the error occurred because of the use of the '*' character in the comment: 5*7=35. Asterisks are used in TRACE to bracket comment sections. If a '*' character is included in a comment, TRACE thinks the comment has ended and TRACE input comes next.

Including an '*' in the description causes the remaining description portion to be interpreted as TRACE input and an error occurs.

The '<PREFIX>.echo' file is a filtered version of the TRACE input file, that strips out comments. The advantage of using the echo file is that it prints the word 'error' on the line the error was found. There are two problems with using the echo file. It strips out comments, so the 'card number' (or line number) where the error occurs will not tend to match the line indicated in the error message. Also SNAP adds comments that help one identify the TRACE input variables. These are stripped out of the echo file. Because of this, the TRACE input file is used instead of the echo file.

Fixing the First Error

The first error occurred in BREAK component 41 in the description section of the input.

1. In the Model Editor, locate and click on the “Faucet” tab at the bottom of the View window.
2. Locate and click on BREAK component 41  at the exit of the faucet in the View Window, and click  on 

in the Properties Window to edit the description comment. You can delete the comment, or replace the '*' with an 'x' to fix the problem. Click 

to save the corrected description.
3. Submit the model again (Refer to the “Input Debugging Exercise Setup” section). Another fatal error should occur.

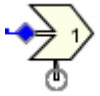
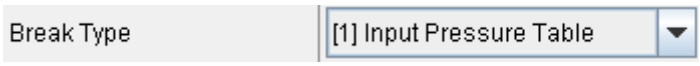
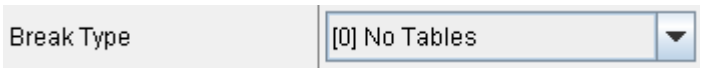
Fixing the Second Error

The screen output in the Job Status window shows TRACE failed on a bad input for BREAK 1.

```
#################
## Input Error ##
#################

 Input error message for break component = 1
error in table specifications. *RBreak*
break parameter table input option ibty = 1 has an independent-variable id number of ibsv = 0
```

The problem is an error in the BREAK table specification. The BREAK parameter table input option ibty = 1, but the independent-variable id number ibsv is 0. From the TRACE input manual (or from SNAP), the variable IBTY is defined as the break type option and IBSV is defined as the BREAK table independent variable ID. From the input error message given in the screen output file (see above), IVTY is set to 0. A BREAK type of 1 indicates a table of pressure versus an independent-variable is to be input. However, the input for IBSV is 0, indicating that no table is input. It turns out that for the Faucet model, it is not IBSV that is incorrect, but rather the BREAK type. BREAK 1 models the cold water input to the faucet. A constant pressure at the cold water inlet is the desired response. To fix the error, do the following:

1. In the Model Editor, click on BREAK 1  which is connected to the bottom right of the faucet PIPE.
2. In the Property Window, change  to  so that BREAK 1 will not require a pressure table input.

3. Submit the model again (Refer to the "Input Debugging Exercise Setup" section). Another fatal error should occur.

Fixing the Third Error

The screen output in the Job Status window shows TRACE failed on a bad input for VALVE 21.

```
          ****************************
          ** Informational Message **
          ****************************

 User information message for valve component = 21
For VALVE = 21 the TRACE internal additive flow loss will be calculated.



          ##################
          ## Input Error ##
          ##################

For VALVE = 22 ivty = 0 and ivsv = -3 and ivsv must be zero.
```

The problem is with VALVE 22. From the TRACE input manual (or from SNAP), the variable IVTY is defined as the valve type and IVSV is defined as the valve table independent variable ID. From the input error message given in the screen output file (see above), IVTY is set to 0. A valve type of 0 indicates a 'constant flow area valve'. However, VALVE 22 is given an IVSV value of -3, indicating that the valve area should be controlled by control block -3. The error message indicates that IVSV should be 0 for a valve type of 0, indicating that there is no control block controlling the valve area. However, in our case, it is not IVSV that is incorrect, but rather the valve type. We would like this to be a controlled valve to adjust the temperature of the water coming out of the faucet. To fix the error, do the following:

4. In the Model Editor, click on VALVE 22  which is connected to the bottom left of the faucet PIPE.

In the Properties Window, hold the mouse over the help icon by Valve Type [0] Constant Flow Area. You should see the tool tip [IVTY] Valve Type. The tool tip shows the input variable name used in the TRACE manual. While the tool tip is visible, drag the mouse over the symbols below it. As the mouse goes over each , a new tool tip should appear indicating the TRACE input variable names. Click on the next to Valve Type [0] Constant Flow Area. A description of the options for the IVTY TRACE input variable is displayed. These tool tips can be very helpful for finding where TRACE input variables are set in SNAP.

5. Change Valve Type [0] Constant Flow Area to Valve Type [1] Flow Area Fraction Table so that VALVE 22 will be a controlled valve rather than a constant area VALVE. Control block -3 (the IVSV control block) will control the flow area fraction of hot water VALVE 22.
6. Submit the model again (Refer to the "Input Debugging Exercise Setup" section). The model should run to completion now.
7. Using AptPlot, verify that the faucet outlet temperature is at the desired value of 105 °F (plot tln-31A12).

Generally, performing a "Check Model" in the SNAP environment (Tools → Check Model) will catch a majority of the input errors that are made. However, there are some errors, like the ones discovered above, that are not detected by SNAP and require an iterative process to locate them using TRACE. Knowing the TRACE nomenclature for the input is important for solving these types of input errors.