



SimSinter gPROMS

Technical Manual

Version 2.0.1

Aug 2019



Copyright (c) 2012 - 2019

Copyright Notice

SimSinter gPROMS was produced under the DOE Carbon Capture Simulation Initiative (CCSI), and is copyright (c) 2012 - 2019 by the software owners: Oak Ridge Institute for Science and Education (ORISE), TRIAD National Security, LLC., Lawrence Livermore National Security, LLC., The Regents of the University of California, through Lawrence Berkeley National Laboratory, Battelle Memorial Institute, Pacific Northwest Division through Pacific Northwest National Laboratory, Carnegie Mellon University, West Virginia University, Boston University, the Trustees of Princeton University, The University of Texas at Austin, URS Energy & Construction, Inc., et al.. All rights reserved.

NOTICE. This Software was developed under funding from the U.S. Department of Energy and the U.S. Government consequently retains certain rights. As such, the U.S. Government has been granted for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable, worldwide license in the Software to reproduce, distribute copies to the public, prepare derivative works, and perform publicly and display publicly, and to permit other to do so.

License Agreement

SimSinter gPROMS Copyright (c) 2012 - 2019, by the software owners: Oak Ridge Institute for Science and Education (ORISE), TRIAD National Security, LLC., Lawrence Livermore National Security, LLC., The Regents of the University of California, through Lawrence Berkeley National Laboratory, Battelle Memorial Institute, Pacific Northwest Division through Pacific Northwest National Laboratory, Carnegie Mellon University, West Virginia University, Boston University, the Trustees of Princeton University, The University of Texas at Austin, URS Energy & Construction, Inc., et al. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the Carbon Capture Simulation Initiative, U.S. Dept. of Energy, the National Energy Technology Laboratory, Oak Ridge Institute for Science and Education

(ORISE), TRIAD National Security, LLC., Lawrence Livermore National Security, LLC., the University of California, Lawrence Berkeley National Laboratory, Battelle Memorial Institute, Pacific Northwest National Laboratory, Carnegie Mellon University, West Virginia University, Boston University, the Trustees of Princeton University, the University of Texas at Austin, URS Energy & Construction, Inc., nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

You are under no obligation whatsoever to provide any bug fixes, patches, or upgrades to the features, functionality or performance of the source code ("Enhancements") to anyone; however, if you choose to make your Enhancements available either publicly, or directly to Lawrence Berkeley National Laboratory, without imposing a separate written license agreement for such Enhancements, then you hereby grant the following license: a non-exclusive, royalty-free perpetual license to install, use, modify, prepare derivative works, incorporate into other computer software, distribute, and sublicense such enhancements or derivative works thereof, in binary and source code form. This material was produced under the DOE Carbon Capture Simulation.

Table of Contents

SimSinter gPROMS	1
1.0 Reporting Issues	1
2.0 Version Log.....	1
3.0 Introduction.....	1
3.1 Motivating Example	1
3.2 Software Requirements for Using SimSinter with gPROMS	2
3.3 What a User Needs to Know About Using gPROMS with SimSinter	2
4.0 Tutorial.....	3
4.1 Configuring gPROMS to Work with SimSinter	3
4.2 Exporting an Encrypted Simulation to Run with SimSinter	12
4.3 Configuring SimSinter to Work with gPROMS	15
4.4 Running gPROMS Simulations with SimSinter	26
5.0 USAGE Information.....	32
5.1 Support.....	32
5.2 gPROMS Input Variable Assignment Types	32
5.3 Parenthesis at the End of Input Variable Reads in gPROMS	34
5.4 SimSinter Cannot Parse Models or Variable Types from Add-On Libraries such as PML.....	35
5.5 Solution Parameters gPLOT is REQUIRED in the Process	38
5.6 SimSinter Cannot use Multi-Dimensional Arrays as Inputs or Outputs.....	38
5.7 Variable and Parameter Defaults Defined in gPROMS	38
5.8 gO:Run_XML License Required	38
5.9 Simulations are Configured with .gPJ Files, but Run with .gENCRYPT	39
5.10 The Name of the gENCRYPT File is Based on the Project File Name	39
6.0 Debugging	39
6.1 How to Debug by Yourself	39
6.2 Known Issues	41
6.2.1 License issue, sim doesn't run	41
6.2.2 ERROR: "gPROMS executable gO:Run_XML.exe could not be found"	42
6.2.3 goORUN_xml produces "Unable to obtain license from server" but runs the simulation.....	42
6.2.4 The Simulation seems to have Succeeded, but all the Output Variables are '0'	43
6.3 How to get help	43
6.4 Reporting Issues	44

List of Figures

Figure 1: Opening the example file.	3
--	---

Figure 2: The BufferTank model.....	4
Figure 3: SimulateTank example process.	5
Figure 4: Right-click and copy the original process.....	6
Figure 5: Making a new process.	6
Figure 6: Rename the process to something useful.	7
Figure 7: SimulateTank_tutorial process before any changes.....	7
Figure 8: SinterConfigGUI Variable Configuration window, Preview Variable frame.	8
Figure 9: SimulateTank_tutorial with all inputs set.	9
Figure 10: gPLOT must be ON in the SOLUTIONPARAMETERS	10
Figure 11: Test edits to the SimulateTank_tutorial process.....	11
Figure 12: Select “Export.”.....	12
Figure 13: Select “Encrypted input file” and then click “OK.”	13
Figure 14: Export entity window.....	14
Figure 15: Start menu, SinterConfigGUI.....	15
Figure 16: SimSinter Configuration File Builder splash screen.....	15
Figure 17: SinterConfigGUI Open Simulation window.	16
Figure 18: SimSinter Save Location.....	17
Figure 19: Additional simulation files may be attached here.....	18
Figure 20: SinterConfigGUI Variable Configuration Page window.	19
Figure 21: SinterConfigGUI automatically discovers input variables from gPROMS.	20
Figure 22: Preview the SimulateTank_tutorial.T101.FlowOut variable.....	21
Figure 23: FlowOut as an output variable.....	22
Figure 24: Additional output variables.....	23
Figure 25: Changed the names of the output variables.....	24
Figure 26: Set defaults for input variables.	25
Figure 27: Vector Default Initialization window.	25
Figure 28: Simulation directory before configuring for runs.	26
Figure 29: Copy the .gENCRYPT file from the input directory.	26
Figure 30: Copy the .gENCRYPT file up to the SimSinter directory. The input directory may then be deleted.	26
Figure 31: Open the Start menu, type “cmd,” and then press “Enter.”.....	27
Figure 32: Change the directory to the user’s simulation directory.	28

Figure 33: Running the DefaultBuilder.	28
Figure 34: The BufferTank_inputs.json file has been created.	29
Figure 35: Running ConsoleSinter.	29
Figure 36: Use Notepad to read the Sinter output file.	30
Figure 37: The outputs from running BufferTank_FO with default inputs.	31
Figure 38: Creating and Setting the connecting parameters	36
Figure 39: Assigning the values of the connecting parameters to the library condenser variables.	37
Figure 40: Declaing output connecting variables	37
Figure 41: Connecting the output variables	38
Figure 42: SOLUTIONPARAMETERS gPLOT := ON is required	38
Figure 43: Launching a command prompt.....	40
Figure 44: Change Directory to the simulation directory	40
Figure 45: Checking for the sinterInput.xml file.....	41
Figure 46: No valid gO:Run_XML license.....	42

List of Tables

Table 1: Foreign Object Method Types Reference Table	34
--	----

To obtain support for the products within this package, please send an e-mail to
ccsi-support@acceleratecarboncapture.org.

SimSinter gPROMS

1.0 REPORTING ISSUES

To report an issue, please send an e-mail to ccsi-support@acceleratecarboncapture.org.

2.0 VERSION LOG

Product	Version Number	Release Date	Description
SimSinter gPROMS	2.0.1	08/15/2019	License update (no functional changes)
SimSinter gPROMS	2.0.0	03/31/2018	Initial Open Source release
SimSinter gPROMS	2016.04.00	04/20/2016	2016 April Release – SinterConfig file format 0.3
SimSinter gPROMS	2015.10.00	11/20/2015	2015 November IAB Release – Added Steady State gPROMS support.
SimSinter gPROMS	2015.6.00	06/30/2015	2015 June Incremental Release – Added Dynamic ACM support.
SimSinter gPROMS	2014.10.0	10/31/2014	2014 October IAB Release – Added FOQUS Integration.

3.0 INTRODUCTION

This document is a supplement to the SimSinter Technical Manual specifically covering the gPROMS simulator. The document assumes that the reader has read the SimSinter Technical Manual. This additional document was written because gPROMS is significantly different from the other simulators SimSinter supports, and the workflow is significantly different.

3.1 Motivating Example

gPROMS provides tools for doing batch runs of large numbers of gPROMS simulations, in the form of gO:Run and gO:Run_XML, so why is SimSinter necessary for gPROMS?

In fact, SimSinter uses the gO:Run_XML tool provided by PSE for running batches of simulations, but SimSinter provides two additional benefits.

1. SimSinter does simplify the process of running jobs compared to the provided PSE tools. Users do not have to edit or generate their own XML files, for example.
2. SimSinter provides the same interface to gPROMS, Aspen Plus®, and Aspen Custom Modeler® (ACM). This allows users to continue using the same tools across all simulators, and to use the Framework for Optimization and Quantification of Uncertainty and Sensitivity (FOQUS) tool for statistical studies and uncertainty quantification (UQ).

The utility of being able to use multiple simulators with the same tools is represented by the current carbon-capture amine-based adsorber and regenerator modelling projects. The carbon-capture system was originally modeled in ACM. Some issues were identified in the ACM version of the simulation. Now that the simulations have been ported to gPROMS the same statistical analyses on both simulators can be run to help uncover problems and increase simulation fidelity.

3.2 Software Requirements for Using SimSinter with gPROMS

SimSinter requires gPROMS 4.0 or newer.

To configure a gPROMS simulation to work with SimSinter, the user must have PSE ModelBuilder 4.0.0 or newer.

To run gPROMS simulations, the simulation machine must have licenses for gO:Run_XML, 4.0 or newer.

However, to configure SimSinter with SinterConfigGUI, no gPROMS licenses of any kind are required, because SinterConfigGUI does not communicate with any gPROMS program, it parses the .gPJ file itself.

3.3 What a User Needs to Know About Using gPROMS with SimSinter

gPROMS and SimSinter interact very differently than SimSinter and Aspen. SimSinter directly communicates with the running Aspen simulation, and can therefore get information on every variable in the simulation at will. SimSinter cannot do that with gPROMS, instead, SimSinter must communicate with gPROMS by reading and writing files. This makes the interaction more complex for both gPROMS and SimSinter. gPROMS must be configured to expect input from SimSinter, and SinterConfigGUI must read the gPROMS .gPJ file to find out what inputs to provide. This extra complexity is the reason for this document.

First, the gPROMS simulation must be configured to accept inputs from SimSinter. This is done with the gPROMS FOREIGN_OBJECT interface. The input variables and parameters are defined to accept input from the FOREIGN_OBJECT. SinterConfigGUI can then read the gPROMS .gPJ file, and discover the inputs by finding the parameters and variables that accept inputs from the FOREIGN_OBJECT.

SimSinter uses gPROMS gO:Run_XML to actually run the gPROMS simulation. gO:Run_XML accepts inputs as an XML file, and then provides those inputs to the correct variables via the FOREIGN_OBJECT interface. The results are passed out via another XML file for SimSinter to read.

It is HIGHLY recommended that the user read section 5.0 USAGE Information before attempting to configure their own simulation.

4.0 TUTORIAL

This section consists of a series of tutorials for every step of configuring gPROMS and SimSinter to work together. All the tutorials are required to have a working simulation run. They are divided up to make it easier for the user to find the section they are interested in.

Before attempting to configure your own gPROMS simulation, it is **HIGHLY** recommended that you read Section 5.0 : Usage Information. It contains all the tips and tricks for handling issues you are likely to run into.

4.1 Configuring gPROMS to Work with SimSinter

Description

Unlike Aspen, changes have to be made to the gPROMS simulation process to work with SimSinter. In fact, SimSinter does not define the inputs to the simulation, gPROMS does. On the other hand, gPROMS does not determine the outputs, SimSinter does. This odd and counter-intuitive situation is the result of how gPROMS gO:Run_XML is designed.

The modification to the gPROMS simulation must be done by a developer with an intimate understanding of the simulation, usually the simulation writer. In some cases additional variables may need to be added to handle an extra step between taking the input and inserting it into the variable where gPROMS will use the data.

1. Open the gPROMS simulation file (ends in “.gPJ”) in ModelBuilder 4.0 or newer. For this example the gPROMS install test file “BufferTank_FO.gPJ” is used. Double-clicking the “.gPJ” file will open ModelBuilder.

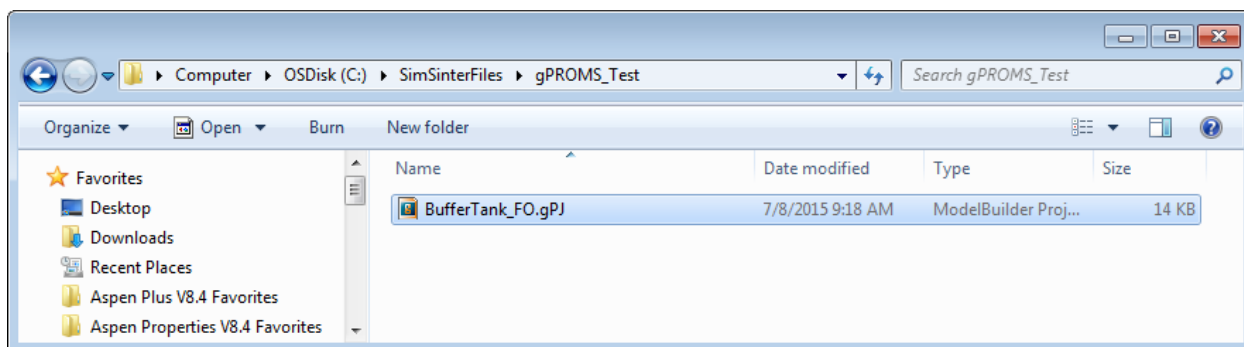


Figure 1: Opening the example file.

2. This simulation was originally a simple BufferTank simulation. However, it was modified into an example of all the different kinds of variables a user can pass in to gPROMS via SimSinter. Therefore, it has a lot of extra variables that do not really do anything, with very generic names, like “SingleInt.”

The simulation consists of a single model, BufferTank, that contains all the simulation logic, and most of the parameter and variable declarations.

The SimSinter simulation will change some of these PARAMETERS and VARIABLES to change the output of the simulation.

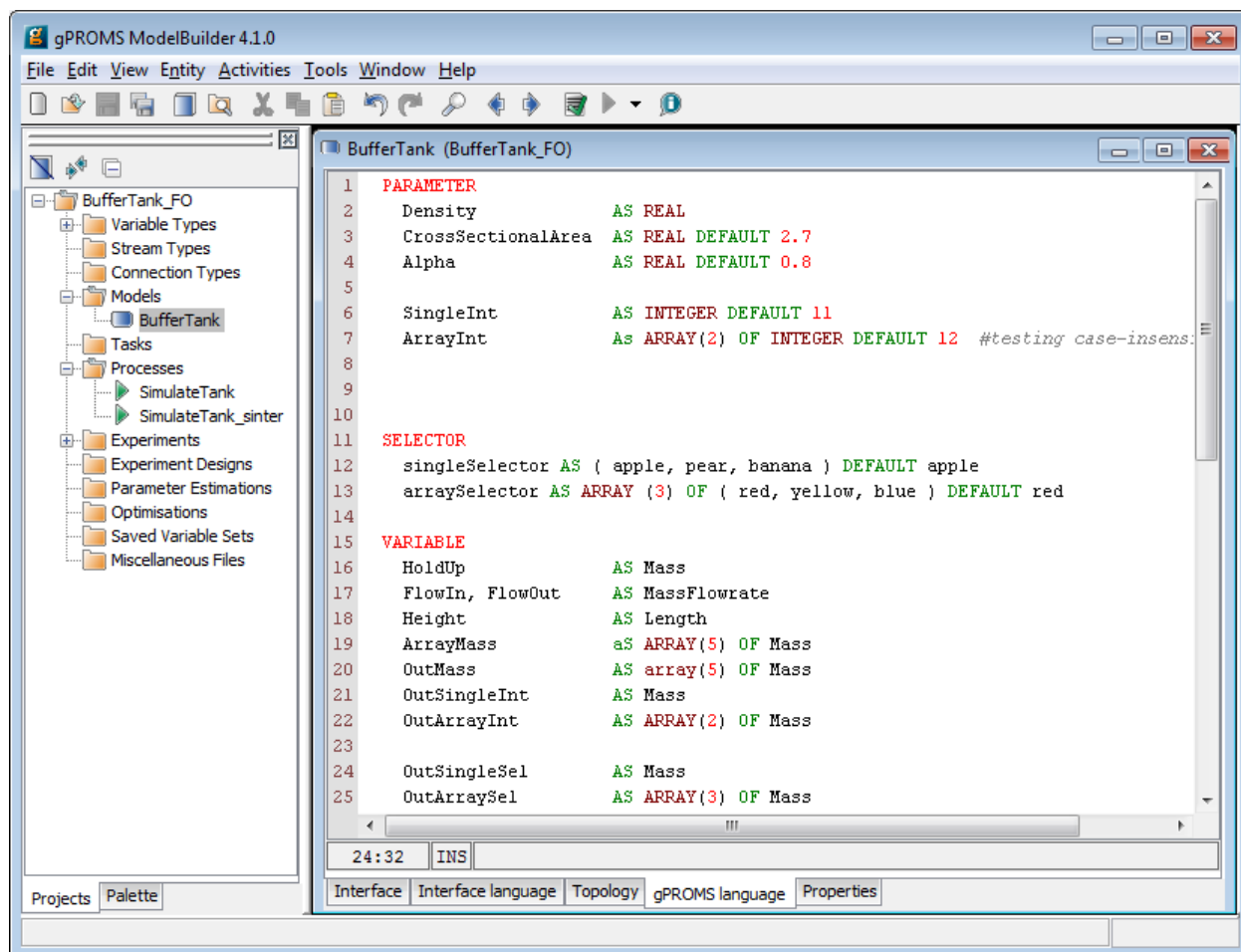


Figure 2: The BufferTank model.

3. The example file contains two processes. SimSinter can only run gPROMS Processes, so any gPROMS simulation must be driven from a process.

SimulateTank is the original BufferTank example with hardcoded values, SimulateTank_Sinter contains the example of setting values with Sinter. The SimulateTank_Sinter example will be recreated in this tutorial.

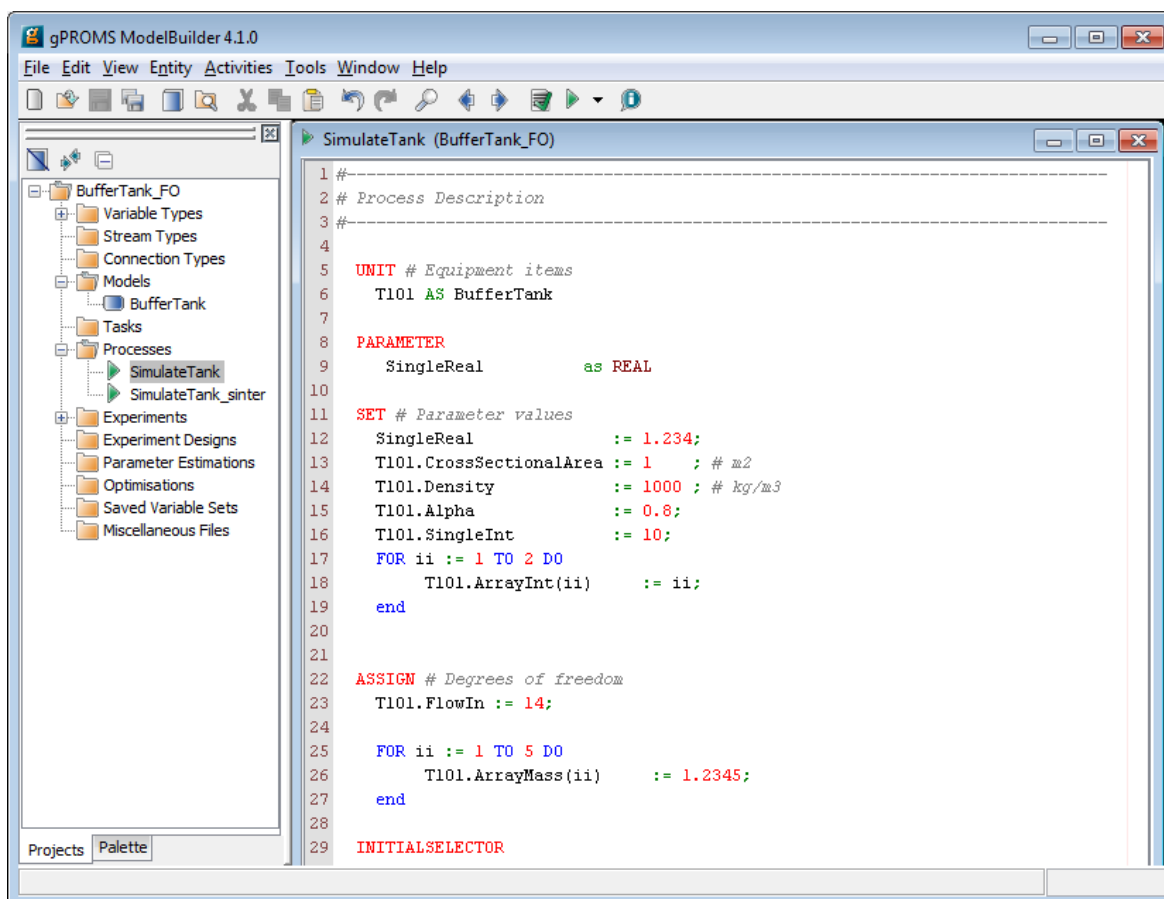


Figure 3: SimulateTank example process.

4. Copy the existing hard-coded process “SimulateTank.”

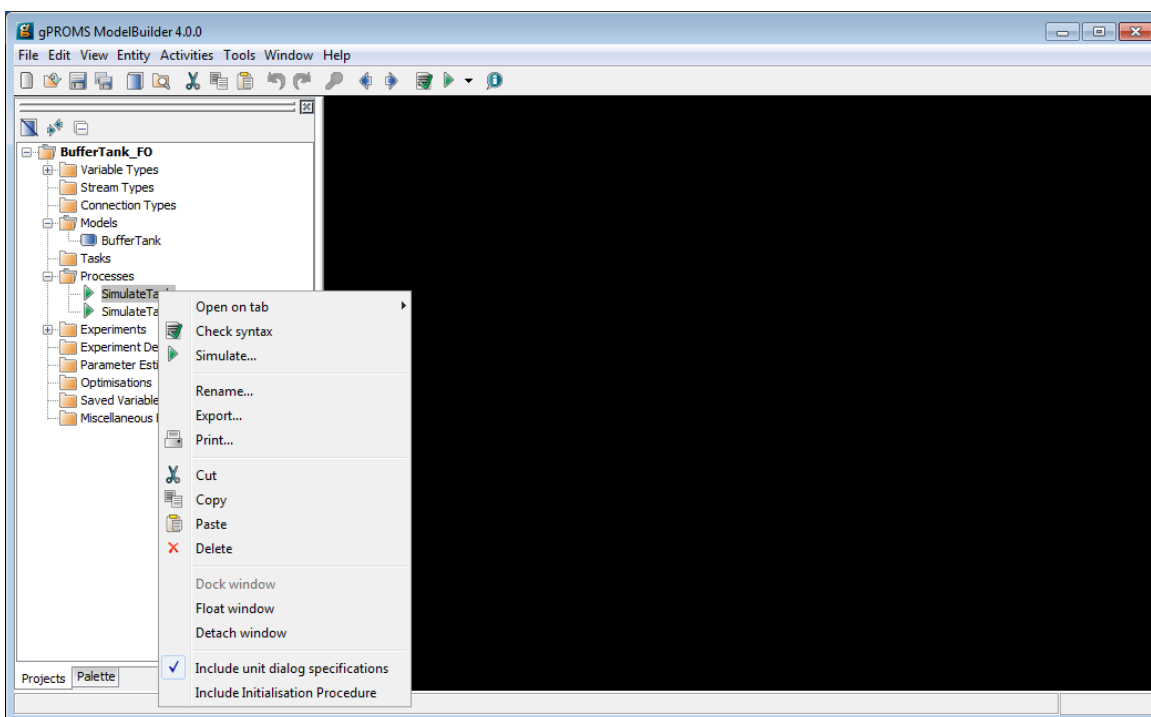


Figure 4: Right-click and copy the original process.

5. Right-click “Processes” and then select “Paste” to create a new process.

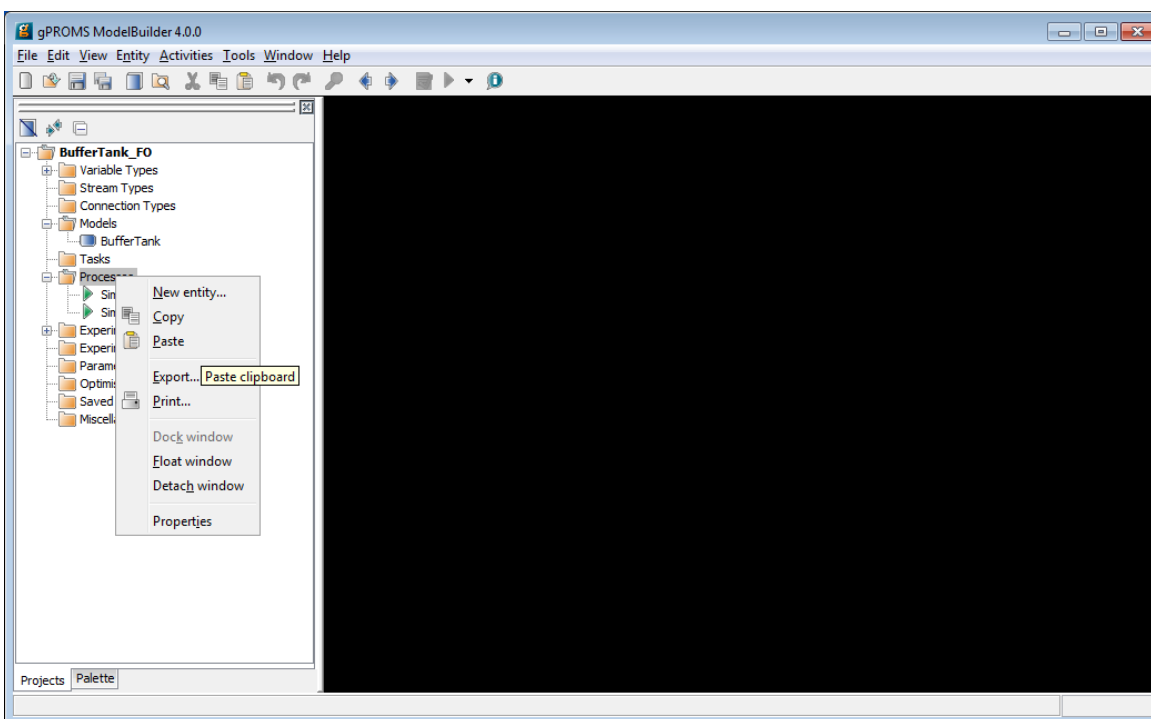


Figure 5: Making a new process.

- The new process will be named “SimulateTank_1.” Rename the process to “SimulateTank_tutorial” by right-clicking “SimulateTank_1” and then selecting “Rename.”

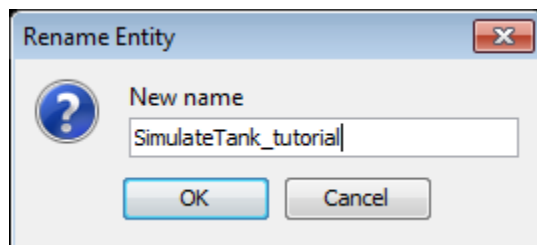


Figure 6: Rename the process to something useful.

- Open the new “SimulateTank_tutorial” process. It has the same hard-coded values as “SimulateTank.”

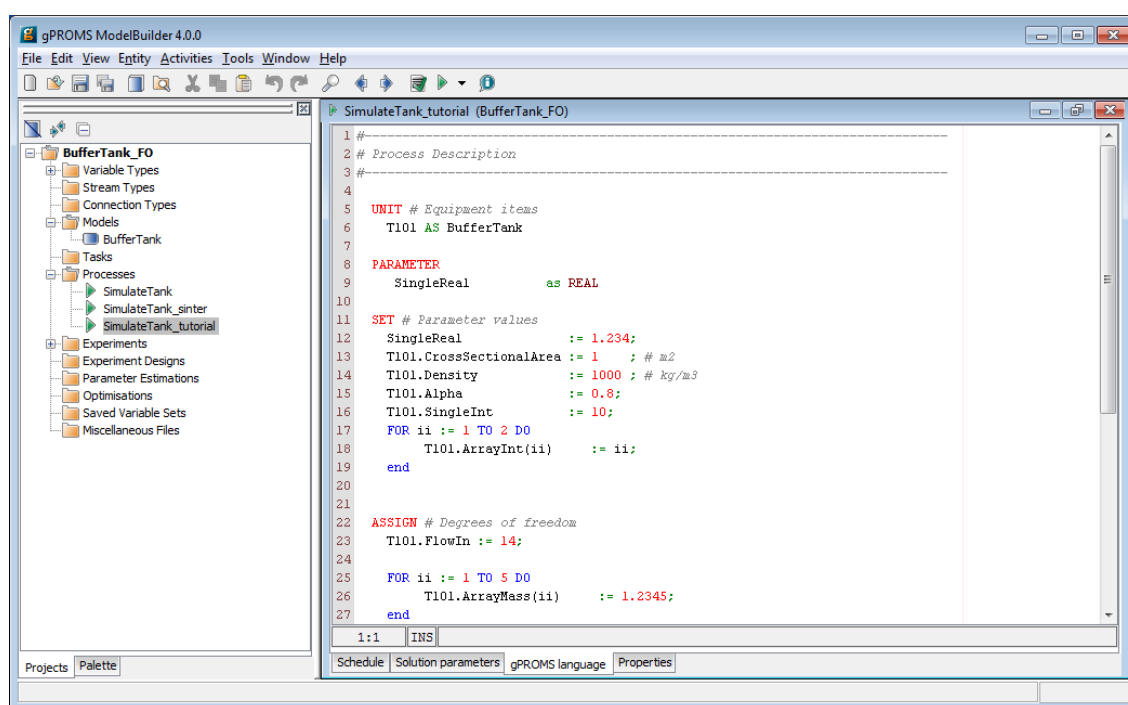


Figure 7: SimulateTank_tutorial process before any changes.

8. Add a FOREIGN_OBJECT, named “FO,” in the PARAMETER section.
Set that FOREIGN_OBJECT to “SimpleEventFOI::dummy” in the SET section.
This FOREIGN_OBJECT is how the user gets inputs from SimSinter.

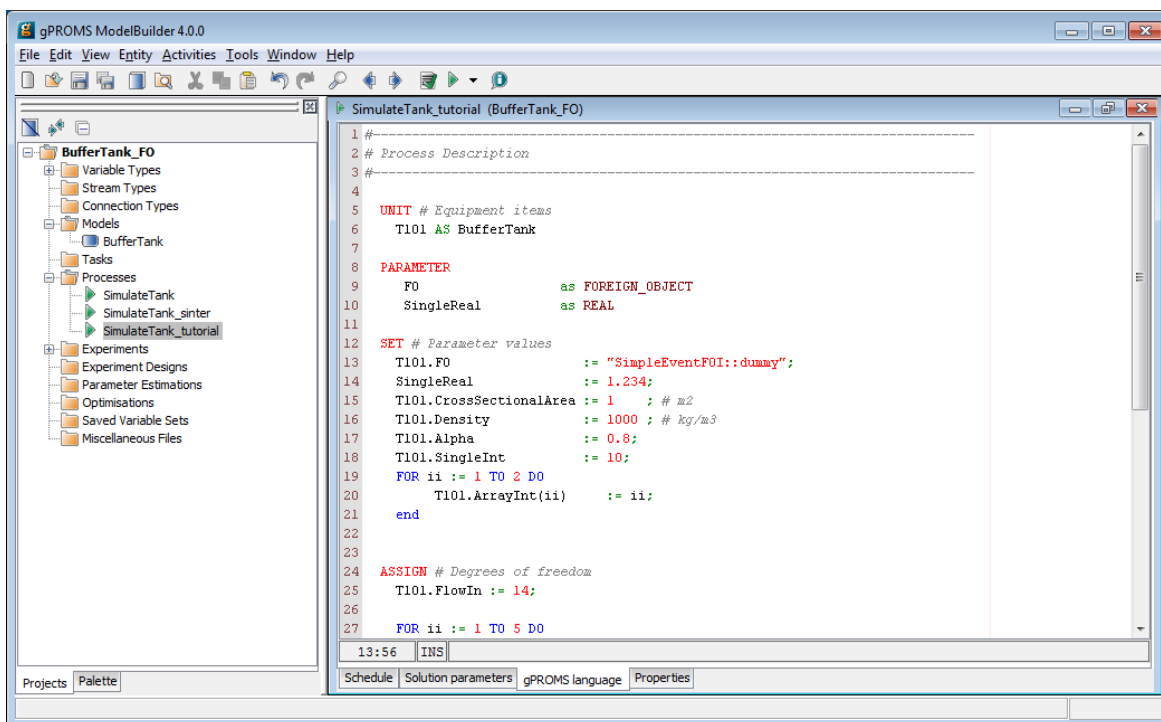


Figure 8: SinterConfigGUI Variable Configuration window, Preview Variable frame.

9. This particular simulation has a large number of pointless input variables simply to demonstrate how to set different types. These are named based on their type. Any variable named similarly to “SingleInt” or “ArraySelector” can be safely ignored for this tutorial. For a full list of the methods for setting different types see Section 5.2 gPROMS Input Variable Assignment Types.

Any variable in the simulation can be an input, whether it is defined in the process or one of the models referenced by the process, or in a model referenced by a model, etc.

All inputs take their values from the FOREIGN_OBJECT the user defined, a period, the type name, two underscores, the input variable name, an open parenthesis, an optional index variable (for arrays), a close parenthesis, and a semicolon.

For a scalar:

```
FO.<Type>__<InputName>();
```

SimSinter can only handle arrays inputted in FOR loops such as:

```
FOR ii := 1 TO <array size> DO
  <ArrayName>(ii) := FO.<Type>1__<InputName>(ii);
END
```

For this example the user needs to set “T101.Alpha” in “PARAMETER,” “T101.FlowIn” in “ASSIGN,” and “T101.Height” in “INITIAL.”

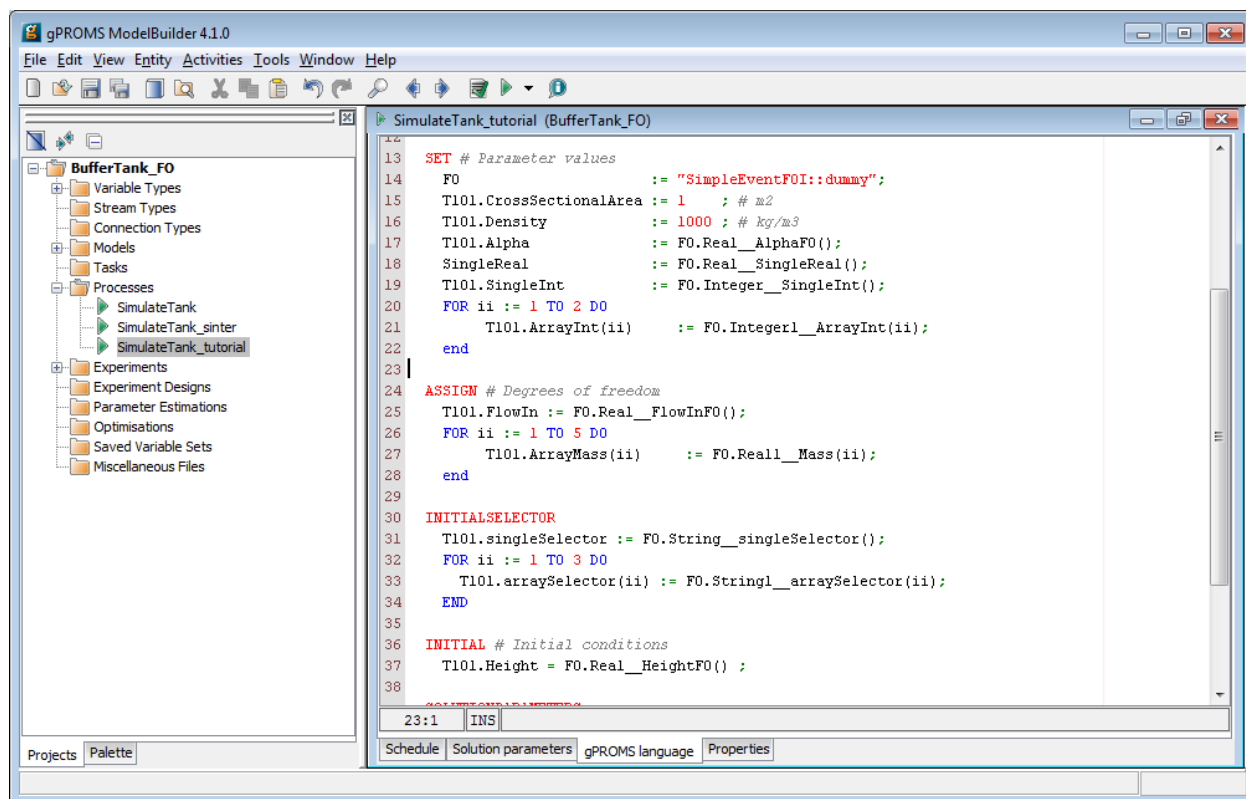


Figure 9: SimulateTank_tutorial with all inputs set.

10. Set `gPLOT := ON` in the `SOLUTIONPARAMETERS` section

Finally, SimSinter will be unable to get any outputs from `gO:Run_XML` if `gPLOT` is not set `ON` in the `SOLUTIONPARAMETERS` section. See Figure 10.

```
334 SOLUTIONPARAMETERS
335     gPLOT := ON
336     InitialisationProcedure := OFF
337
```

Figure 10: gPLOT must be ON in the SOLUTIONPARAMETERS

11. Test the “SimulateTank_tutorial” by selecting it and then clicking the “green Simulate triangle”. When the simulation runs it will ask for every input variable the user has defined.

For the example variables that do not effect the simulation, such as “SingleInt”, any valid value is acceptable.

For the values that do effect the simulation, these values work:

- REAL__AlphaFO = .08
- REAL__FlowInFO = 14
- REAL__HeightFO = 7.5

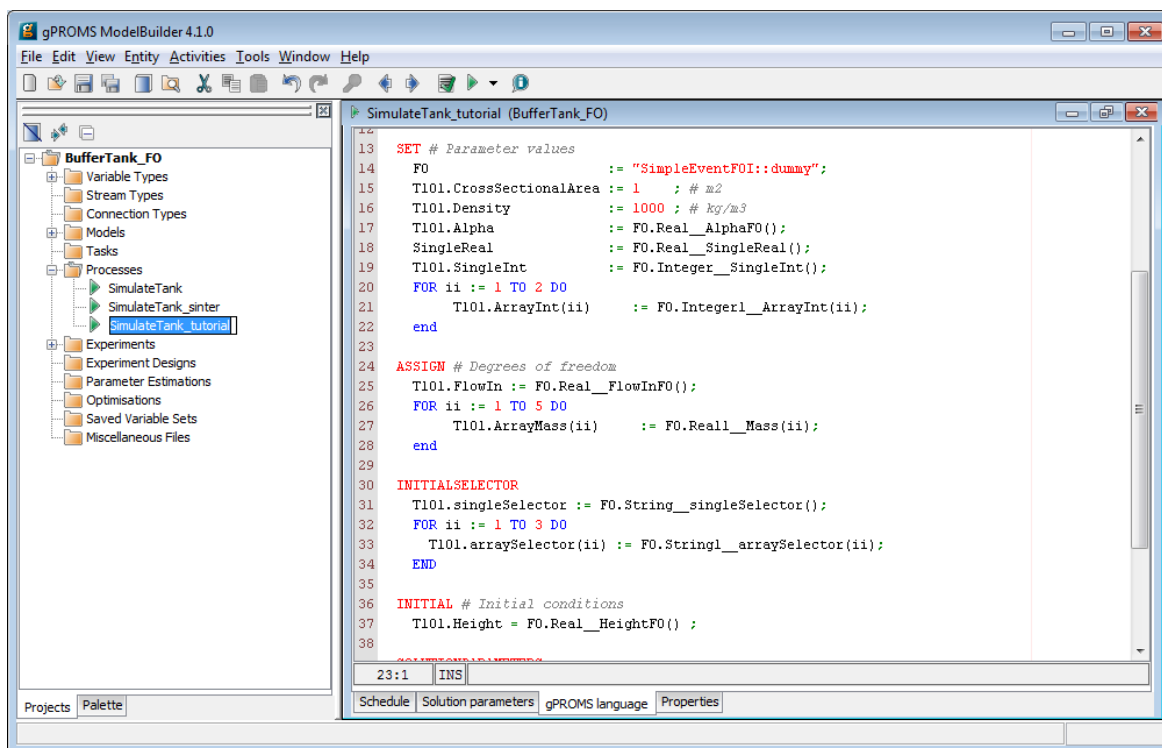


Figure 11: Test edits to the SimulateTank_tutorial process.

4.2 Exporting an Encrypted Simulation to Run with SimSinter

SimSinter can only run encrypted gPROMS simulations. These files have the .gENCRYPT extension. If the user's additions to the simulation for reading input variables ran correctly in the previous section, that process is ready to be exported for use by SimSinter.

1. Right click the “process” to export (“SimulateTank_tutorial”) and then select “Export.”

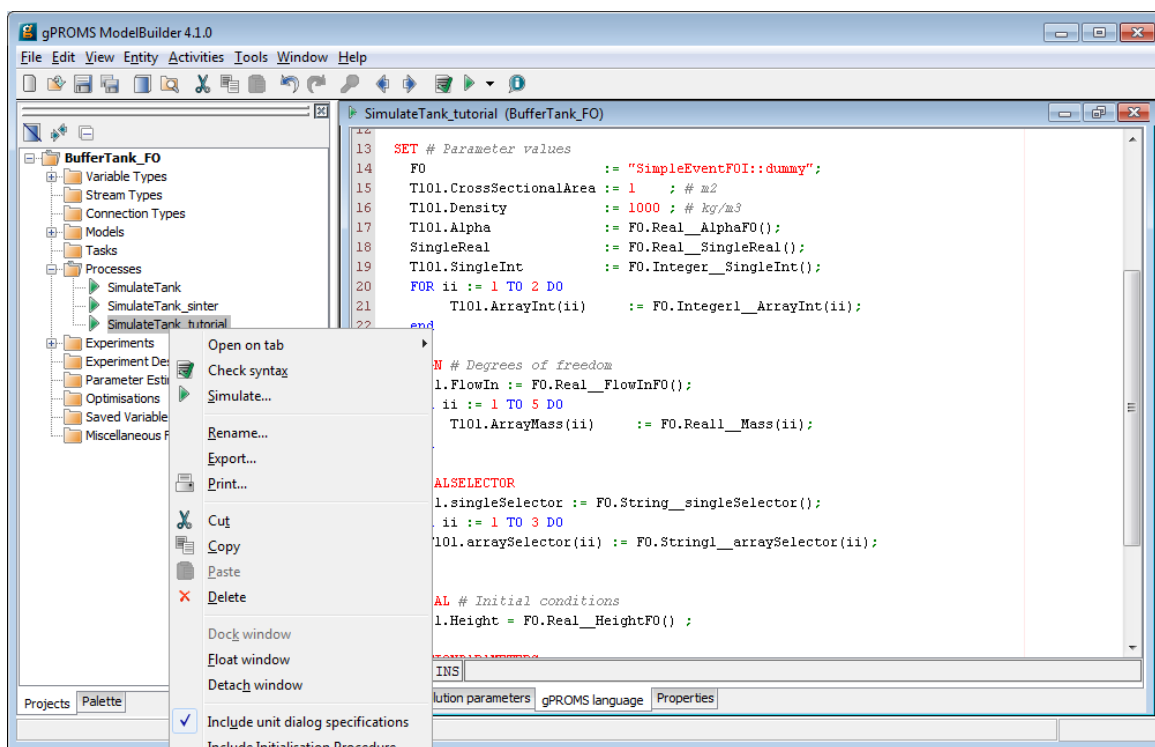


Figure 12: Select “Export.”

2. In the resulting Export window, select “Encrypted input file for simulation by gO:RUN” and then click “OK.”

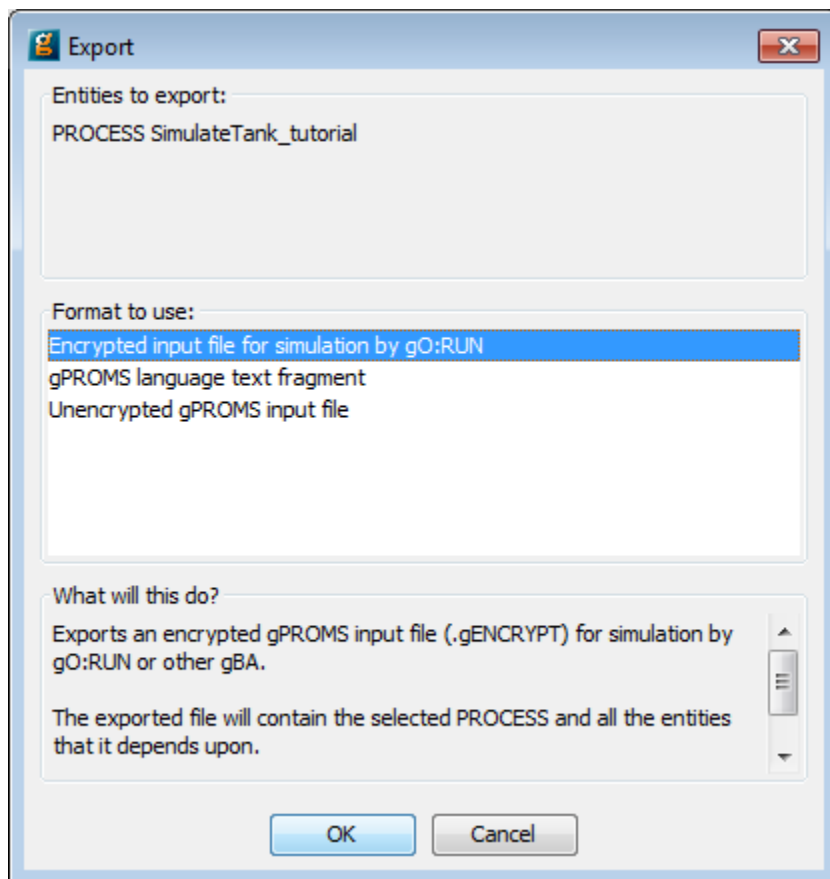


Figure 13: Select “Encrypted input file” and then click “OK.”

- On the second window, be sure to set the “Export directory” to a directory the user can find. Preferably one without any other files in it so the user would not be confused by the output.

If the “Input file name” or “Encryption password” are not changed, SimSinter will be able to guess the password. If either of these values are changed, the user will have to set the correct password in the SinterConfigGUI password setting.

A Decryption password is probably unnecessary, as the user has the original file. SimSinter does not use it.

When finished setting up these fields, click “Export Project.”

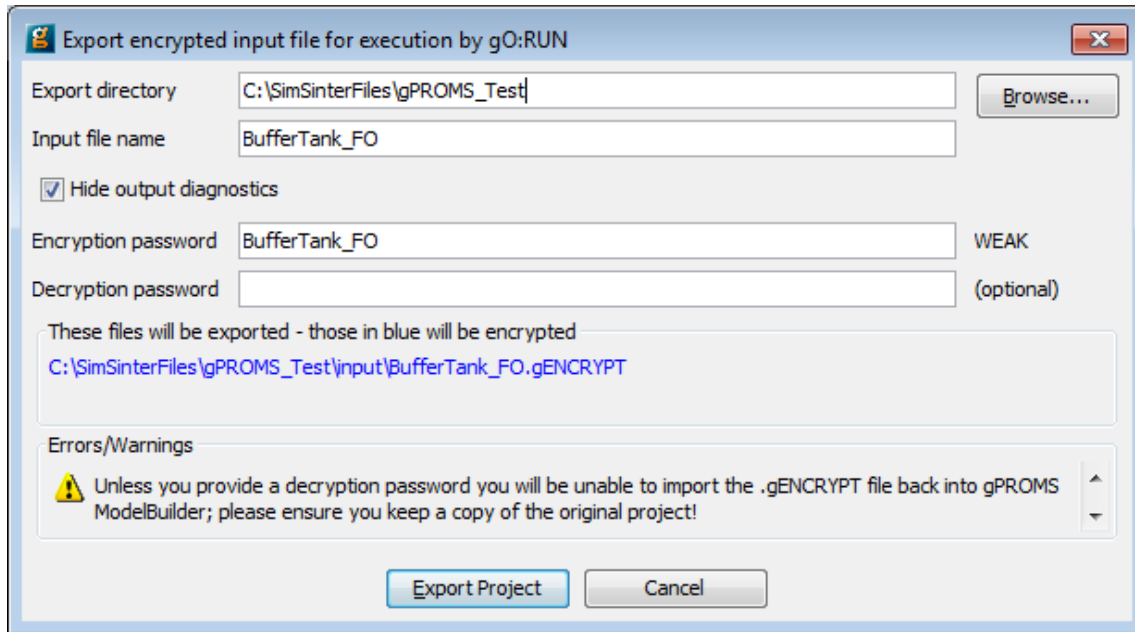


Figure 14: Export entity window.

- The resulting .gENCRYPT file is saved to a subdirectory named “input” in the save directory specified in Step 3. The first part of the name is identical to the .gPJ filename. The user should not rename the file because the SinterConfig file will guess this name, and currently changing it requires editing the SinterConfig file. We recommend that you copy the .gENCRYPT file up to the same directory as the .gPJ file, so that FOQUS can find it.

4.3 Configuring SimSinter to Work with gPROMS

A SimSinter configuration file must also be produced to tell SimSinter how to run the gPROMS simulation.

1. Open the “SinterConfigGUI” from the “Start” menu, as shown in Figure 14.

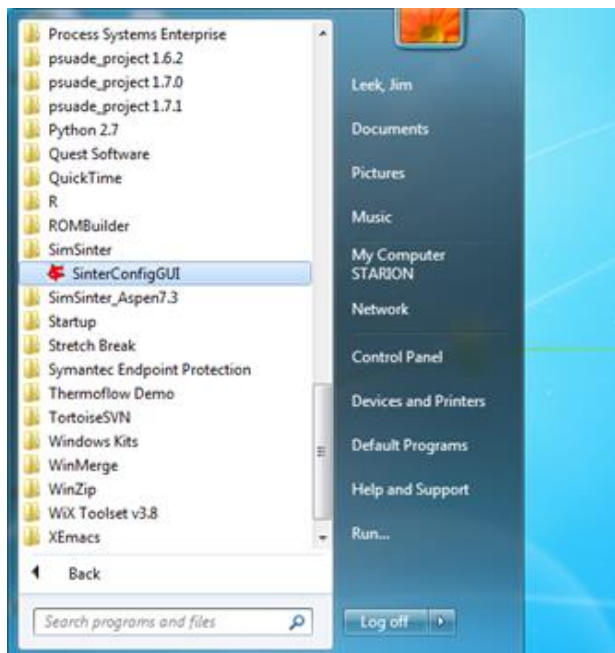


Figure 15: Start menu, SinterConfigGUI.

2. Initially the SimSinter Configuration File Builder splash screen displays, as shown in Figure 15. Either click the “splash screen” to proceed, or wait 10 seconds for the screen to close automatically.



Figure 16: SimSinter Configuration File Builder splash screen.

3. The SinterConfigGUI Open Simulation window displays as shown in Figure 16. Click “Browse” to select the file to open and then click “Open File and Configure Variables” to open the file.

SinterConfigGUI **cannot** read the .gENCRYPT file that is actually run by SimSinter. Instead, the user must open the .gPJ file the ModelBuilder uses.

In this case use the file configured in the 4.1 Configuring gPROMS to Work with SimSinter tutorial. Or the example may be found at: C:\SimSinterFiles\gPROMS_Test\BufferTank_FO.gPJ.

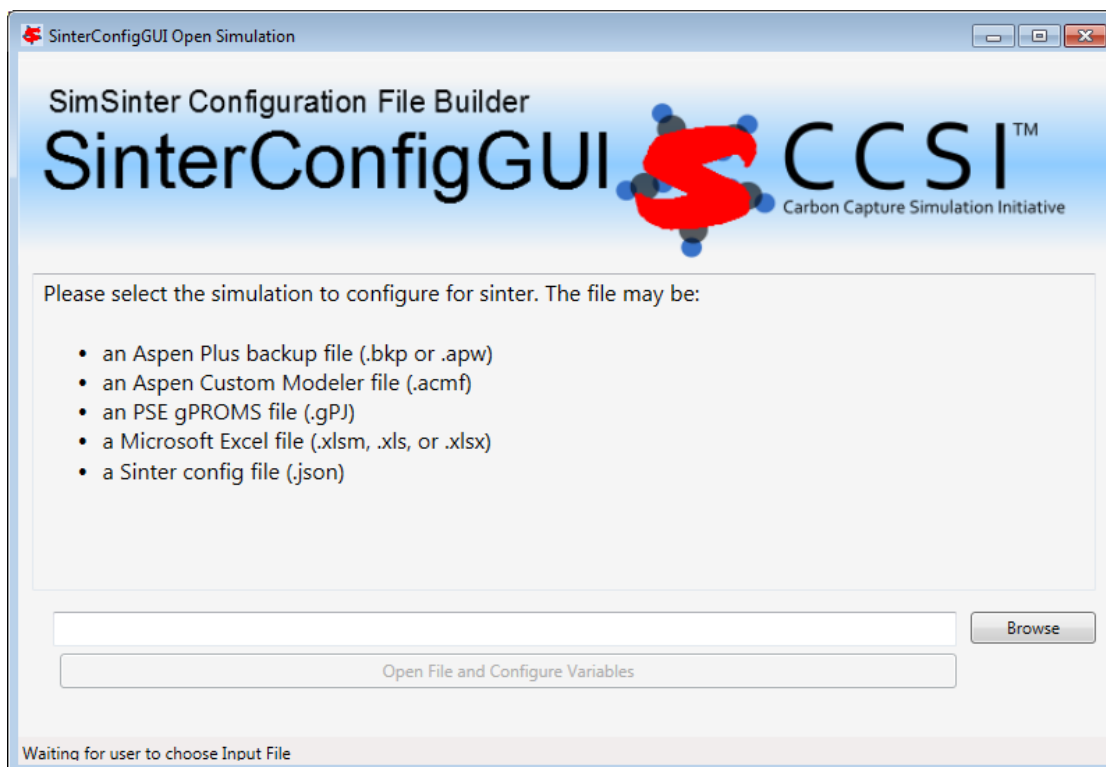


Figure 17: SinterConfigGUI Open Simulation window.

4. The SinterConfigGUI Simulation Meta-Data window displays as shown in Figure 17. Unlike the other simulations, gPROMS has not started up in any way. SinterConfigGUI does not get information from gPROMS directly, it must parse the .gPJ file instead.

5. The first and most important piece of meta-data is the “SimSinter Save Location” at the top of the window. This is where the Sinter configuration file is saved. The system suggests a file location and name. The user should confirm this is the intended location of the files to not accidentally overwrite other files.

Futhermore, the configuration file autosaves when “Next >” is clicked, so please ensure that the filename is correct, and will not overwrite any important files.

The screenshot shows the 'SinterGomgGUI Simulation Meta-Data' window. The 'SimSinter Save Location' field is highlighted with a red circle, displaying the path 'C:\SimSinterFiles\gPROMS_Test\BufferTank_FO.json'. Below this, the 'Application' is set to 'gPROMS', the 'Version' is '4.0.0', and the 'Constraint' is 'AT-LEAST'. The 'Input Files' section lists 'BufferTank_FO.gENCRYPT'. The 'Simulation Meta-Data' section includes fields for 'Title', 'Description', 'Author', and 'Date' (set to '11/17/2015'), along with a 'Version' field set to '0.1'. Navigation buttons '< Back' and 'Next >' are located at the bottom right.

Figure 18: SimSinter Save Location.

6. SimSinter cannot enforce version constraints on gPROMS, so there is no point in setting them, except as a method of informing the user.
- Some simulations have additional files they require to run, but this simulation does not, so a full tutorial will not be given here. For more information see the Dynamic ACM simulation section **Error! Reference source not found.** in the SimSinter Technical Manual.
- If any additional files are required, they may be attached to the simulation via the Input Files section. The simulation file itself is always included in the Input Files, and cannot be removed.

The screenshot shows the 'SinterConfigGUI Simulation Meta-Data' window. The 'SimSinter Save Location' is set to 'C:\SimSinterFiles\gPROMS_Test\BufferTank_FO.json'. The 'Application' is 'gPROMS', 'Version' is '4.0.0', and 'Constraint' is 'AT-LEAST'. The 'Input Files' section, highlighted with a red box, contains a list with 'BufferTank_FO.gENCRYPT' and buttons for 'Add File' and 'Remove File'. The 'Simulation Meta-Data' section includes fields for 'Title', 'Description', 'Author', 'Date' (set to '11/17/2015'), and 'Version' (set to '0.1'). There are also buttons for '< Back' and 'Next >'.

Figure 19: Additional simulation files may be attached here

- The SinterConfigGUI Variable Configuration Page window displays as shown in Figure 19. gPROMS has two settings, ProcessName and password. SimSinter has guessed at both the ProcessName and the password. For this example the password is correct, but the ProcessName is incorrect. SimulateTank is the process that is not configured to work with SimSinter.

On the left side is the Variable Tree. The root is connected to the three processes defined in this .gPJ file.

Change the “ProcessName” to “SimulateTank_tutorial.”

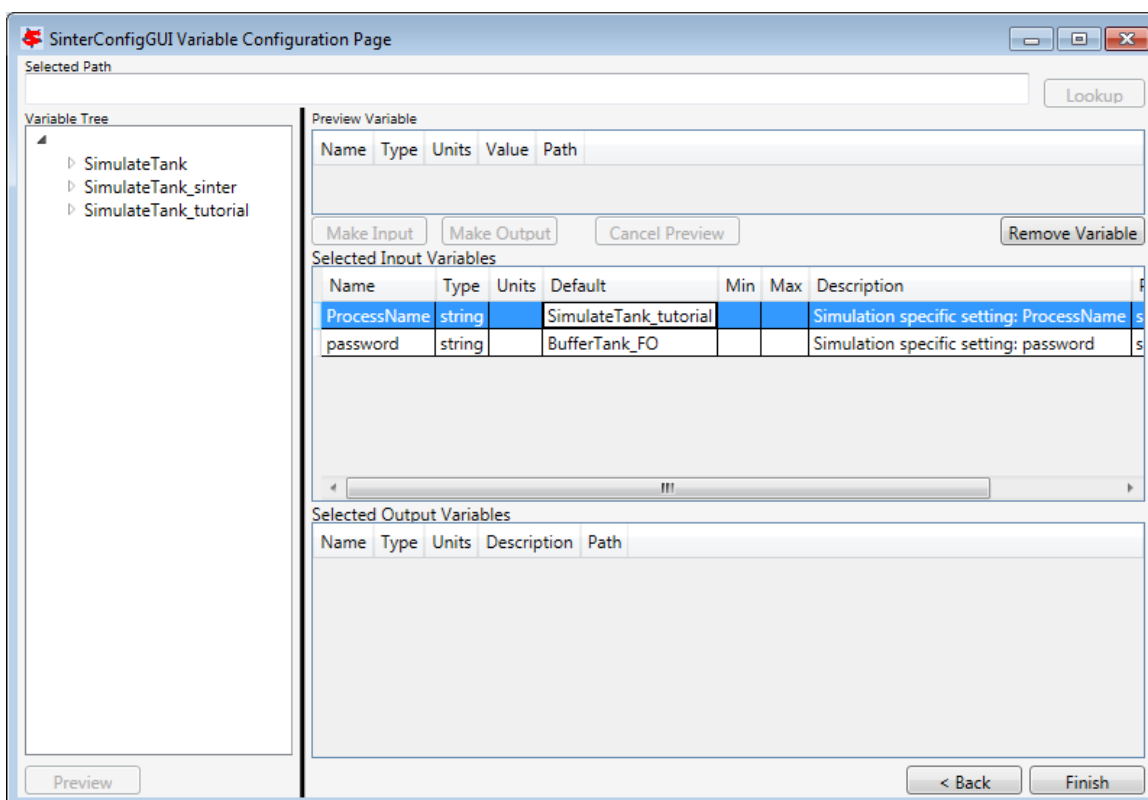


Figure 20: SinterConfigGUI Variable Configuration Page window.

8. When the user changes the ProcessName and presses “Enter” (or click away), the user will see all of the available input variables. This is because the input variables have been configured in gPROMS, and SimSinter has parsed them out of the .gPJ file, as long as the user has the ProcessName set correctly. This also means that the user cannot add new input variables in SinterConfigGUI, only in gPROMS.

SimSinter also does its best to identify the Default values, min, and max of the variables.

The Default can only be calculated from the file if it was defined purely in terms of actual numbers. SimSinter cannot evaluate other variables or functions. So “DEFAULT 2 * 3.1415 * 12” will work. But “DEFAULT 2 * PI * radius” will not work, and SimSinter will just set the default to “0.”

Min and max values are taken from the variable Type, if there is one.

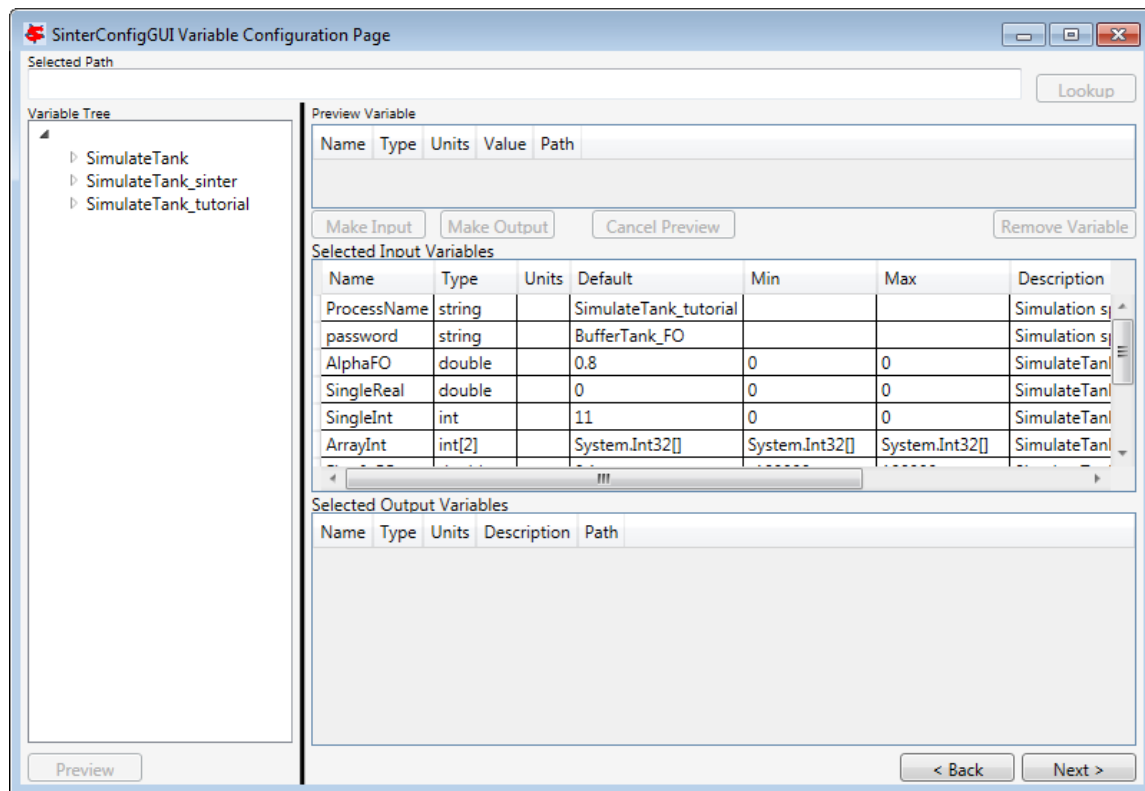


Figure 21: SinterConfigGUI automatically discovers input variables from gPROMS.

9. The output values can now be entered. Expand the “SimulateTank_tutorial” process on the tree, and then expand the “T101” model. Double-click “FlowOut” to make it the preview variable.

Notice that the “Make Input” button is unavailable. As stated above, the user cannot make new input variables in SinterConfigGUI. Only “Make Output” is allowed.

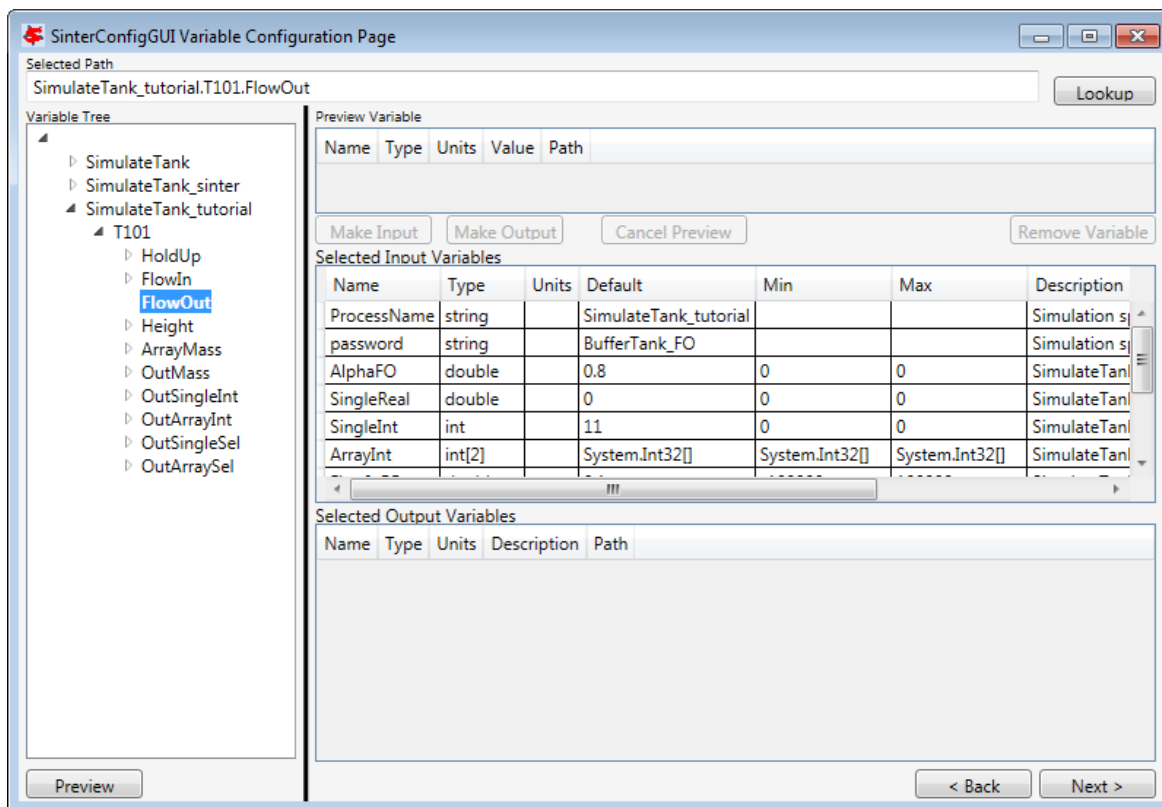


Figure 22: Preview the SimulateTank_tutorial.T101.FlowOut variable.

10. Click “Make Output,” FlowOut is made an output variable as shown in Figure 22.

The user can update the description, but SimSinter made a good guess this time, so there is not any need to.

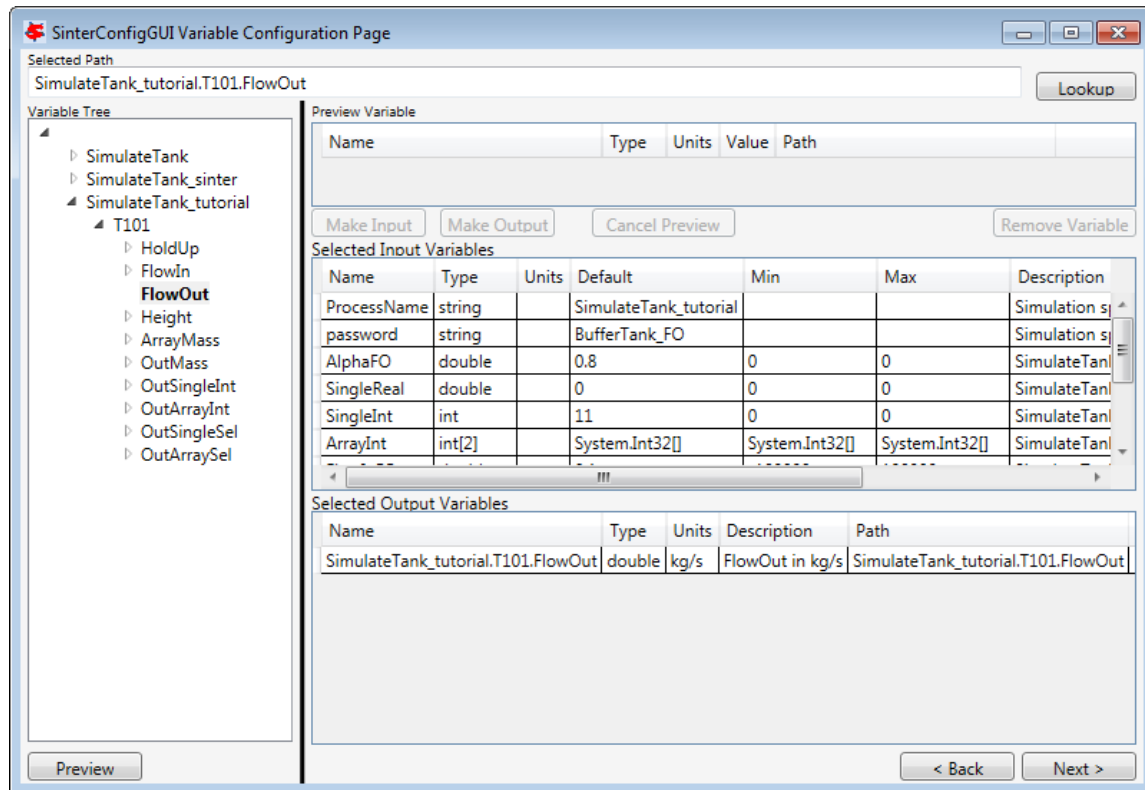


Figure 23: FlowOut as an output variable.

11. By the same method, make output variables “HoldUp” and “Height.”

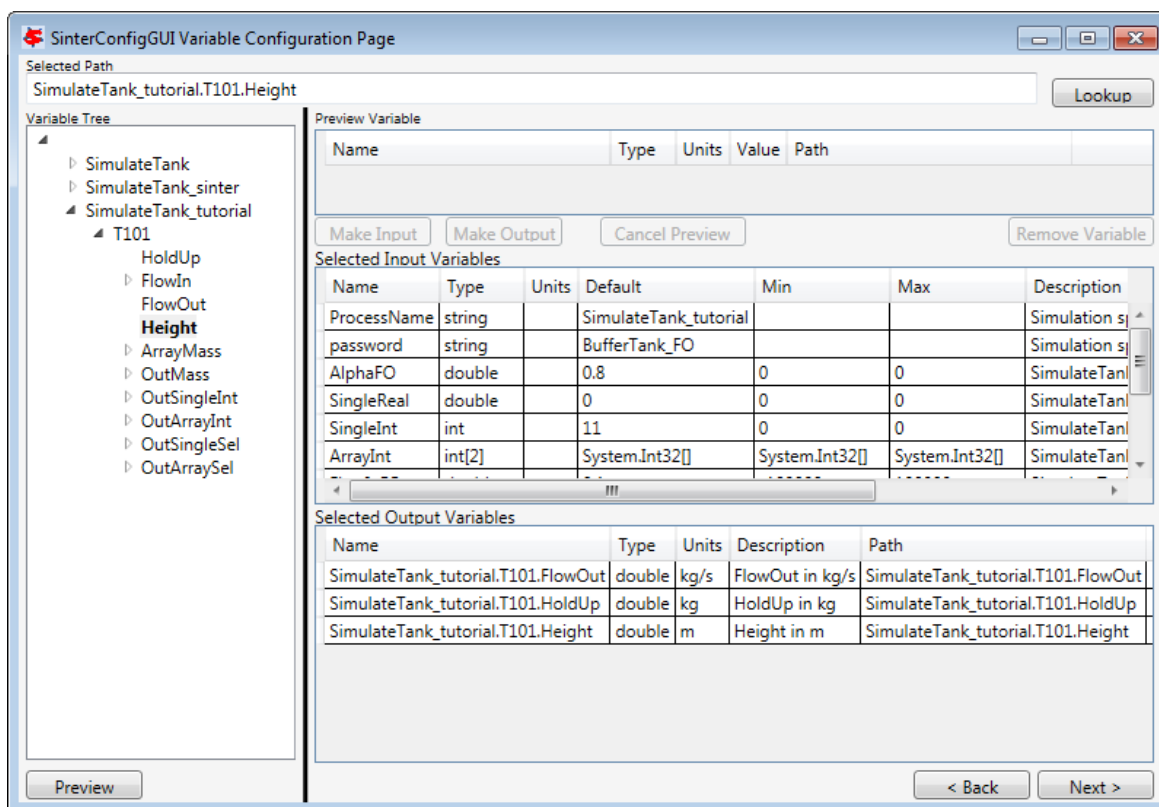


Figure 24: Additional output variables.

12. The variables names should be made shorter. Simply click the “name column” and then change the name to a preferred name.

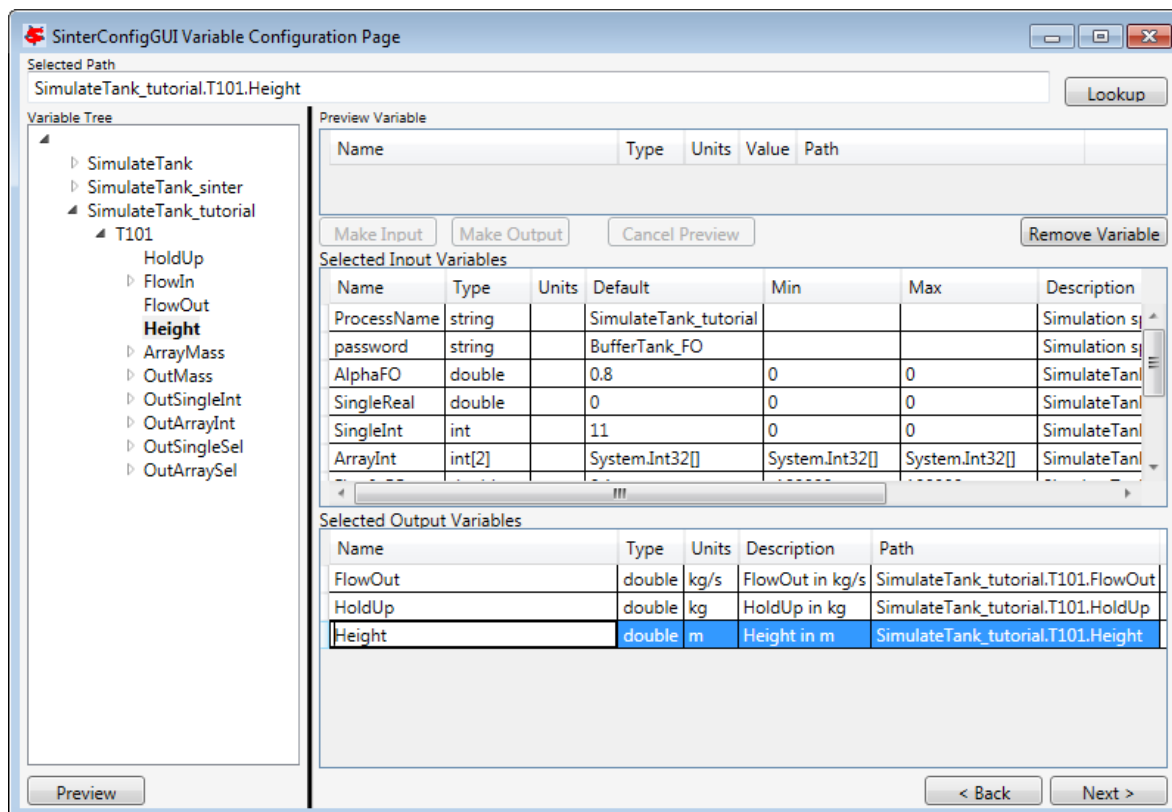


Figure 25: Changed the names of the output variables.

13. For future testing, make sure the defaults are good values. The only three input variables that matter have the following defaults:

- AlphaFO: 0.8
- FlowInFO: 14
- HeightFO: 7.5

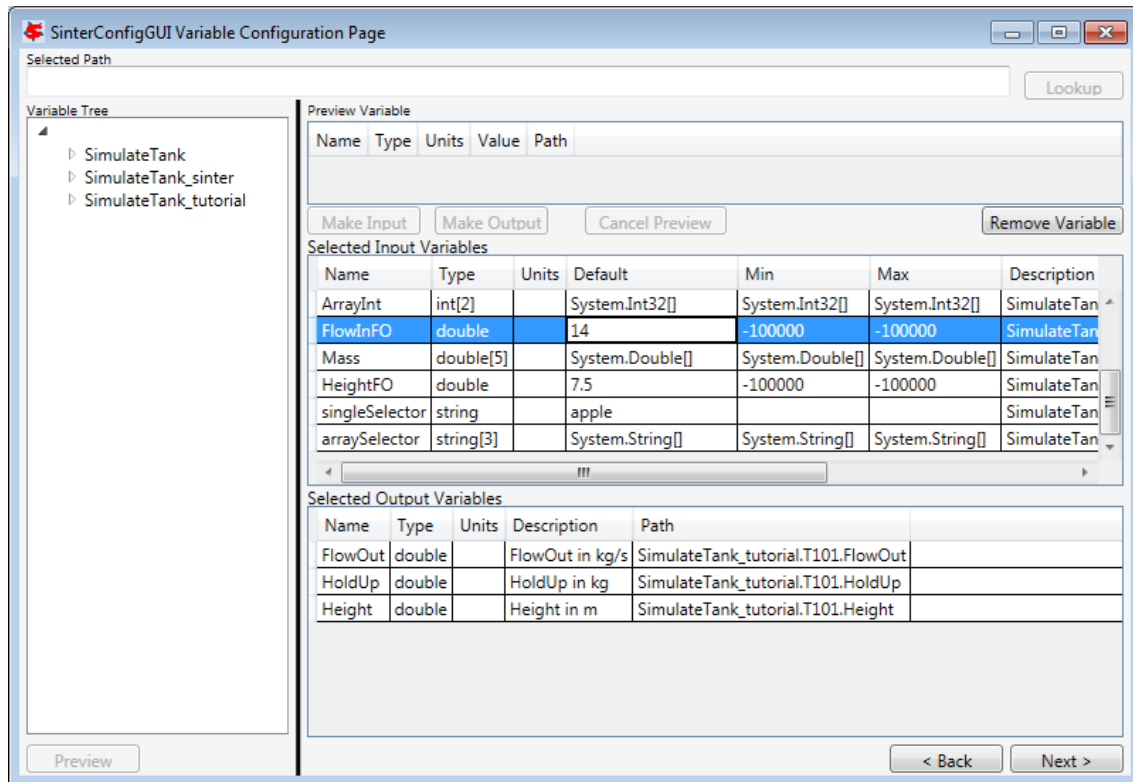


Figure 26: Set defaults for input variables.

14. When finished making output variables, click “Next” at the bottom of the variables window.

If there were any input vectors, the Vector Default Initialization window displays. Here the default values of the vectors can be edited.

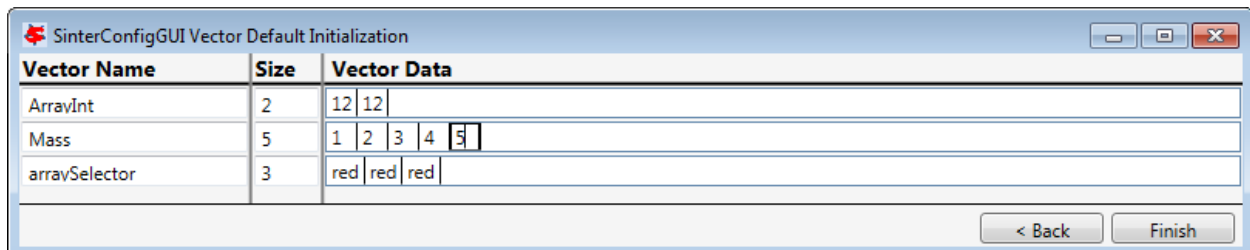


Figure 27: Vector Default Initialization window.

15. Click “Finish” to save the configuration file and then close SimSinter.

4.4 Running gPROMS Simulations with SimSinter

After configuring gPROMS, exporting a .gENCRYPT file, and creating a SinterConfig file, this should be the state of the gPROMS simulation directory:

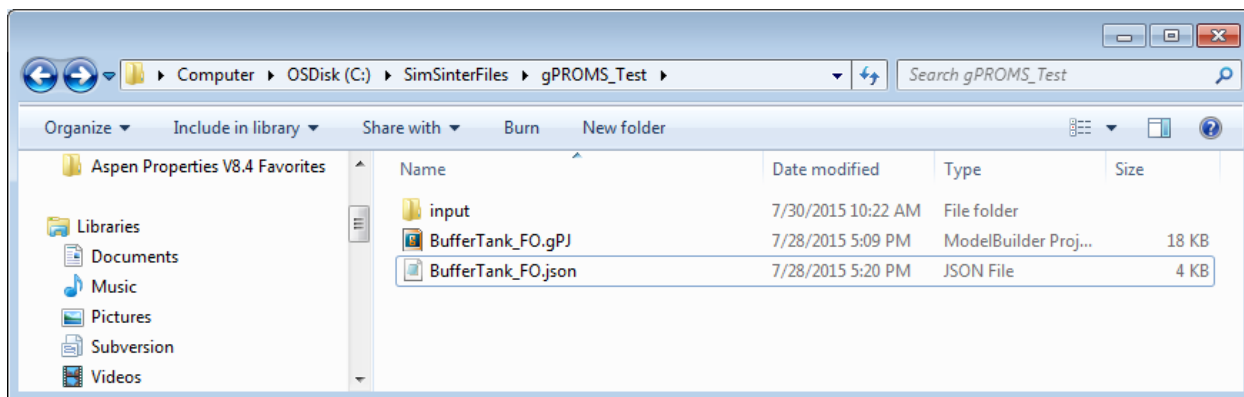


Figure 28: Simulation directory before configuring for runs.

1. The .gENCRYPT file is in the input directory. Move it up to the same level as the SinterConfig file. After which the user may delete the input directory.

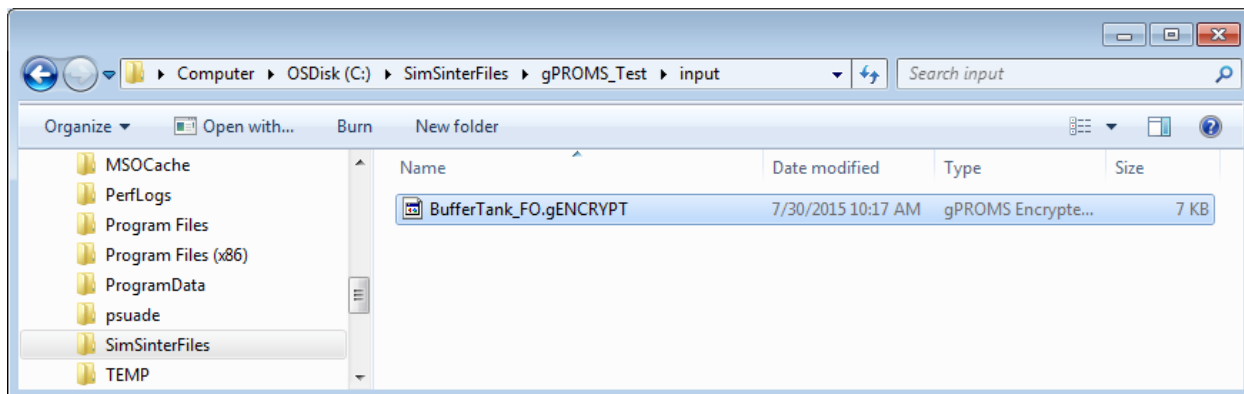
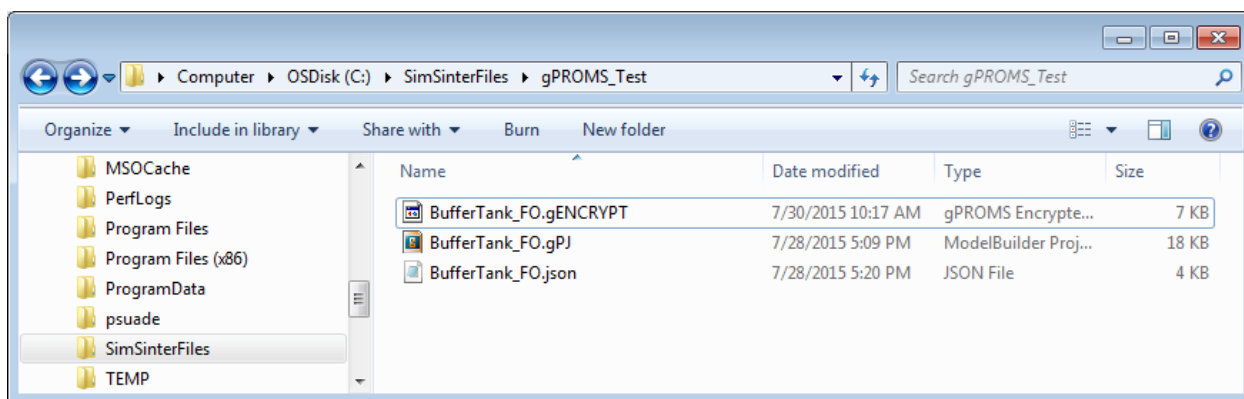


Figure 29: Copy the .gENCRYPT file from the input directory.



**Figure 30: Copy the .gENCRYPT file up to the SimSinter directory.
The input directory may then be deleted.**

- Figure 29 contains three files. To run the simulation only the .json and .gENCRYPT files are required, but to configure the simulation or change it, the .gPJ file is required.

If the user wishes to run the simulation on Turbine, simply upload the .json and .gENCRYPT files there. The .gPJ file may also be included for archival and documentation purposes, but it is not required.

If the user wishes to test the simulation locally first, continue the tutorial.

- To run the simulation locally, a set of inputs is needed to be passed in. These inputs can be generated with the DefaultBuilder.exe program included in the SimSinter distribution.

DefaultBuilder.exe is run from the Windows command line. Open a command line window by clicking “Start” or the “Windows Key” and then typing “cmd.”

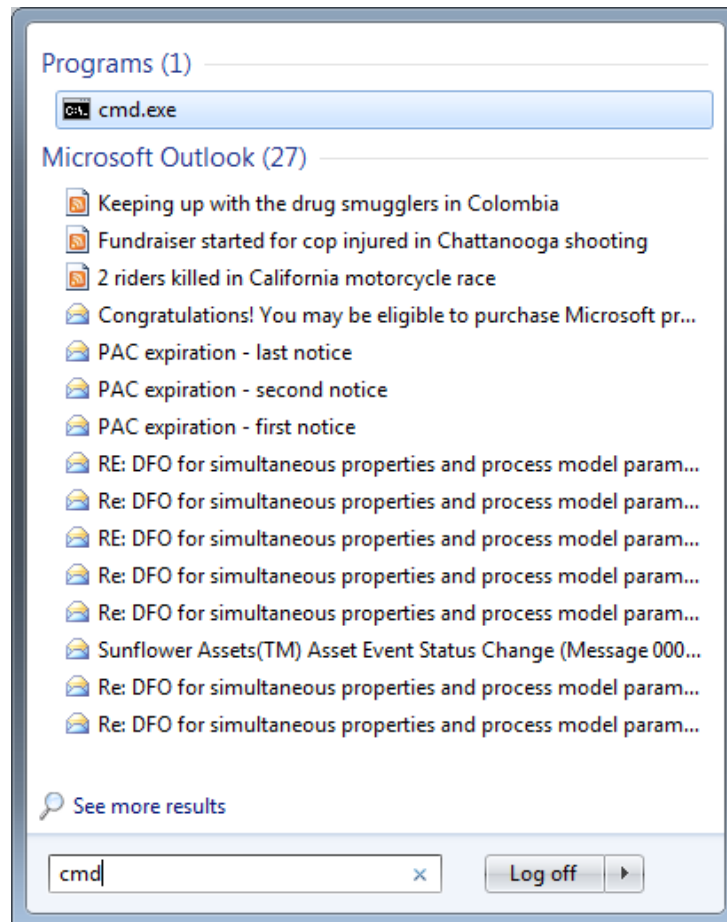


Figure 31: Open the Start menu, type “cmd,” and then press “Enter.”

4. Change the directory to the user's simulation directory. If the user is using the SimSinterFiles test directories, the command is:

```
cd c:\SimSinterFiles\gPROMS_Test
```

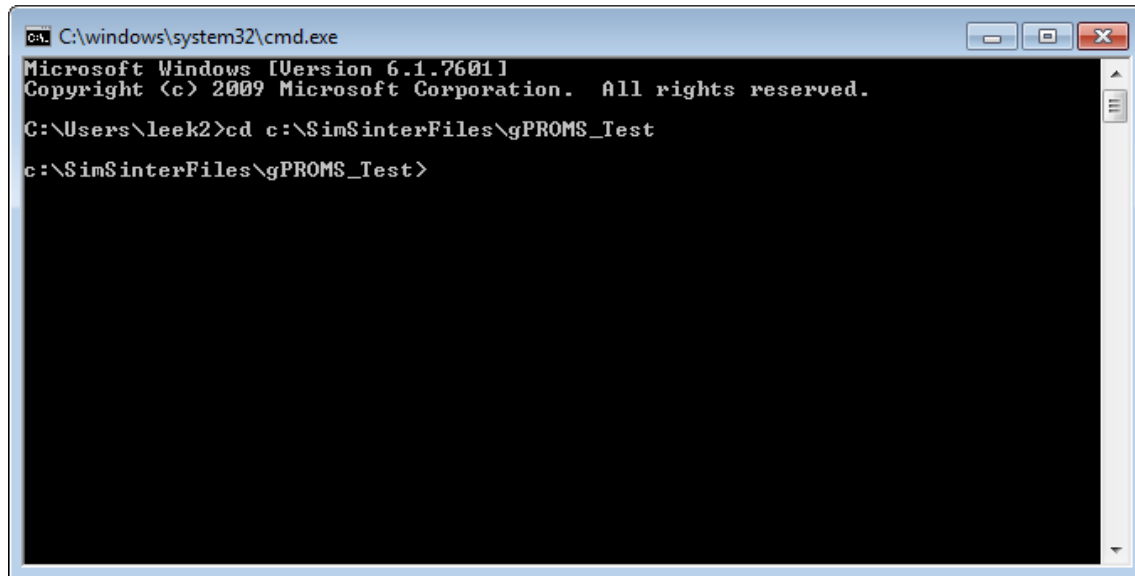


Figure 32: Change the directory to the user's simulation directory.

5. On the "command line" run the DefaultBuilder. It takes two arguments:
 - a. The filename of the SinterConfig File.
 - b. The output filename for the defaults file. Give this a nice descriptive name.

Here is an example:

```
c:\Program Files (x86)\CCSI\SimSinter\DefaultBuilder.exe  
BufferTank_F0.json BufferTank_inputs.json
```

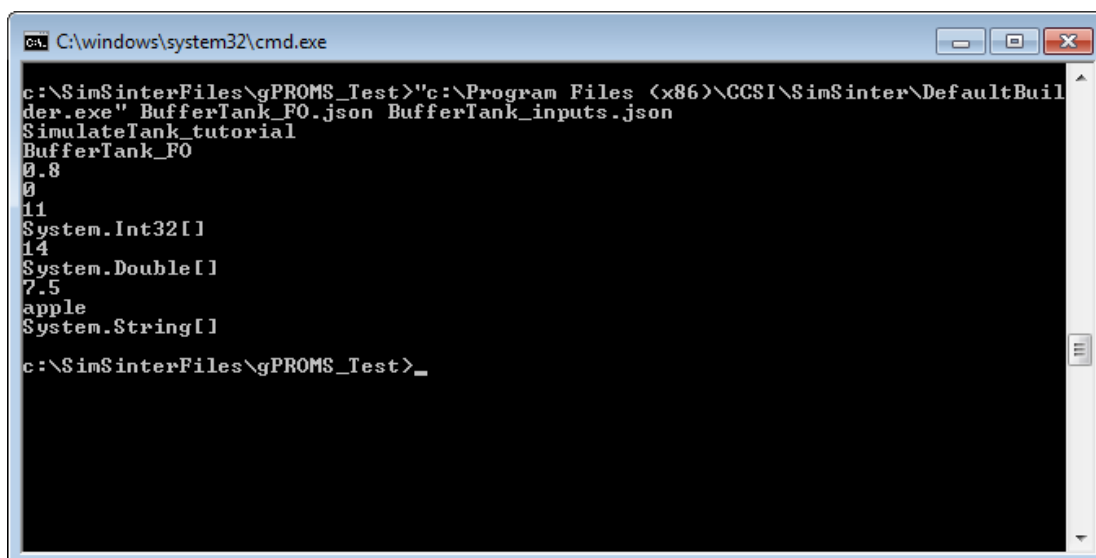


Figure 33: Running the DefaultBuilder.

6. Observe in Windows explorer that the defaults/inputs file has been generated. This file may be edited in Notepad to change the values of the inputs and run different configurations. But for this test it is better to run with the defaults to avoid possible errors.

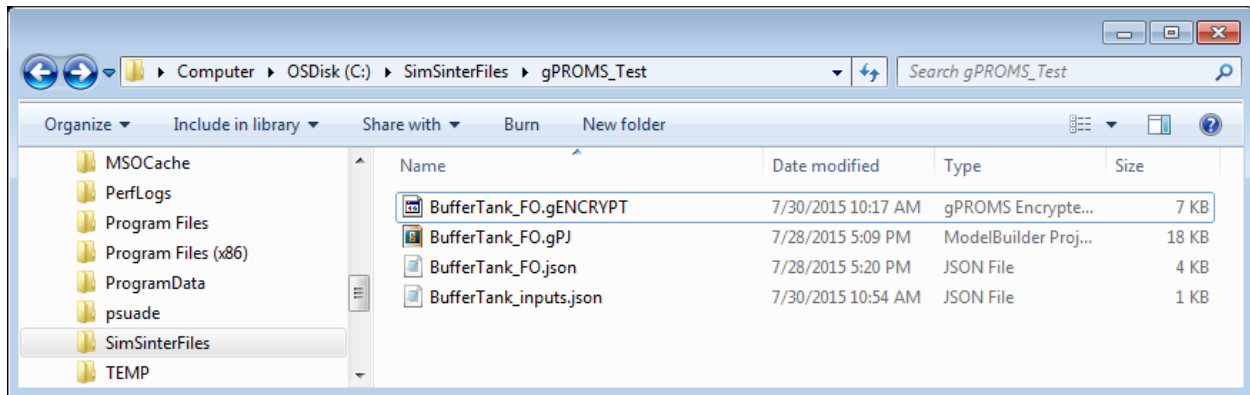


Figure 34: The BufferTank_inputs.json file has been created.

7. SimSinter can be run from the command line with the new input file.

ConsoleSinter takes three arguments:

- a. The SinterConfig file
- b. The inputs file (here the defaults that were generated are used)
- c. A file for the SimSinter outputs

The command is:

```
c:\Program Files (x86)\CCSI\SimSinter\ConsoleSinter.exe
BufferTank_FO.json BufferTank_inputs.json BufferTank_outputs.json
```

If the simulation runs properly, the outputs will be very uninteresting. If there is an error there will be a much longer more complex message output.

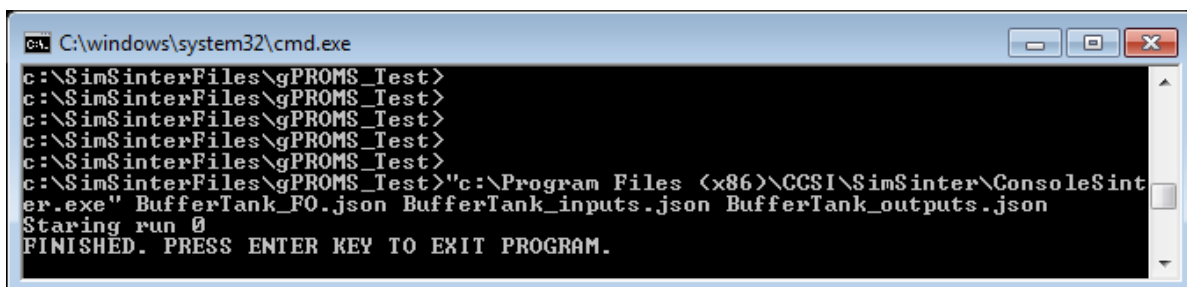
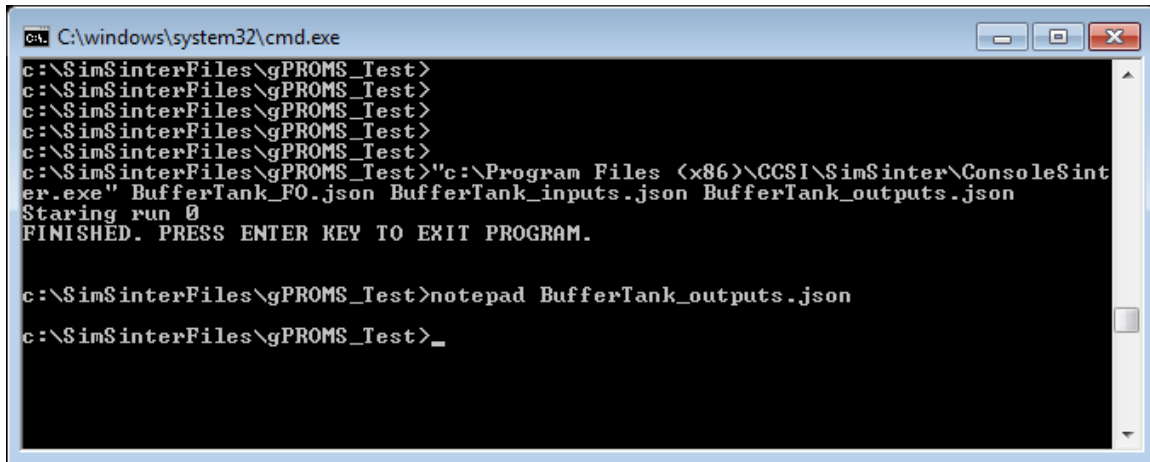


Figure 35: Running ConsoleSinter.

8. To double check that the simulation ran correctly, look at the Sinter outputs in Notepad:

```
notepad BufferTank_outputs.json
```

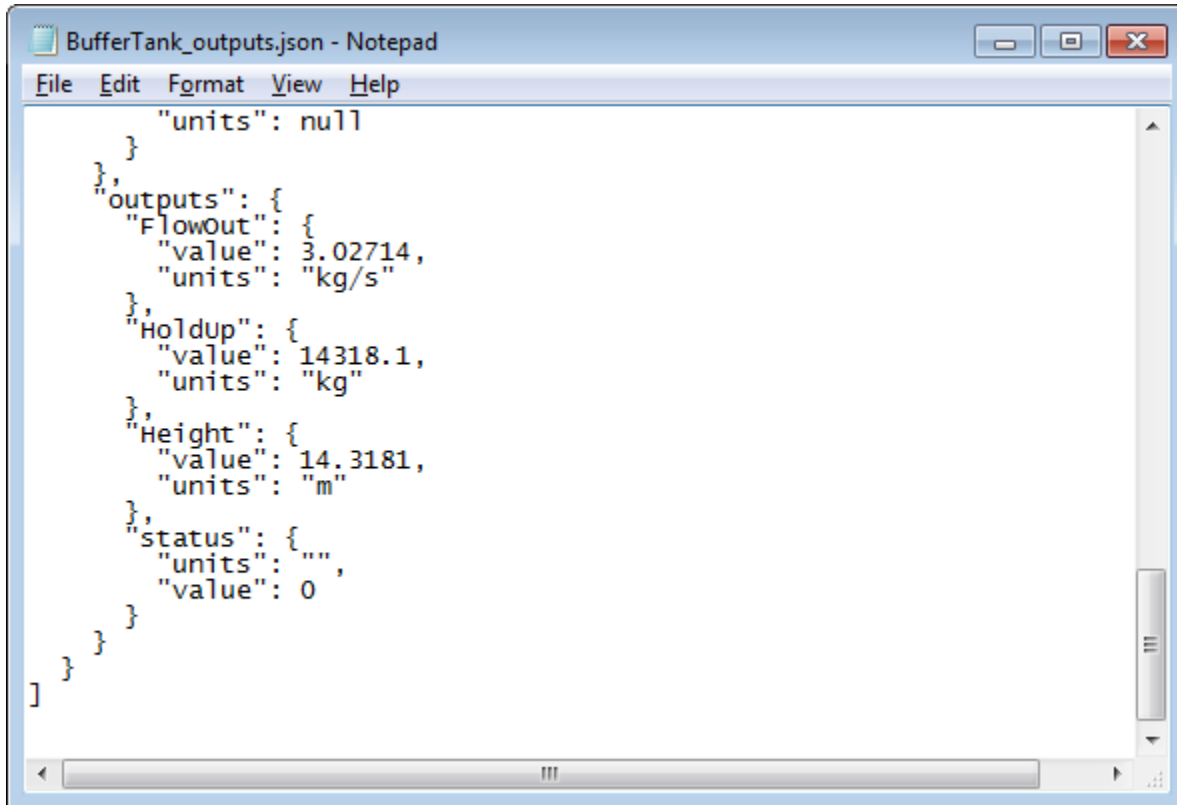


```
C:\windows\system32\cmd.exe
c:\SimSinterFiles\gPROMS_Test>
c:\SimSinterFiles\gPROMS_Test>
c:\SimSinterFiles\gPROMS_Test>
c:\SimSinterFiles\gPROMS_Test>
c:\SimSinterFiles\gPROMS_Test>
c:\SimSinterFiles\gPROMS_Test>"c:\Program Files (x86)\CCSI\SimSinter\ConsoleSinter.exe" BufferTank_F0.json BufferTank_inputs.json BufferTank_outputs.json
Starting run 0
FINISHED. PRESS ENTER KEY TO EXIT PROGRAM.

c:\SimSinterFiles\gPROMS_Test>notepad BufferTank_outputs.json
c:\SimSinterFiles\gPROMS_Test>_
```

Figure 36: Use Notepad to read the Sinter output file.

9. Scroll down to the “outputs” section. The values should be:
- FlowOut: 3.02714
 - HoldUp: 14318.1
 - Height: 14.3181

A screenshot of a Notepad window titled "BufferTank_outputs.json - Notepad". The window contains a JSON object representing simulation outputs. The "outputs" section lists three variables: FlowOut, Holdup, and Height, each with a numerical value and a unit. FlowOut is 3.02714 kg/s, Holdup is 14318.1 kg, and Height is 14.3181 m. There is also a "status" field with a value of 0 and an empty unit string. The JSON is formatted with indentation and line breaks.

```
[{"units": null}, {"outputs": {"FlowOut": {"value": 3.02714, "units": "kg/s"}, "Holdup": {"value": 14318.1, "units": "kg"}, "Height": {"value": 14.3181, "units": "m"}, "status": {"value": 0, "units": ""}}}]
```

Figure 37: The outputs from running BufferTank_FO with default inputs.

5.0 USAGE INFORMATION

The gPROMS and SimSinter interaction is quite complex, and there are a number of details, tips, and tricks that do not cleanly fit into another section. Those issues and tips are listed in this section.

5.1 Support

For support, e-mail the problem with all of the relevant details to:
ccsi-support@acceleratecarboncapture.org.

5.2 gPROMS Input Variable Assignment Types

SimSinter allows three variable types, and vectors of those three types. This section describes the details of using them with gPROMS. All input variables come from a FOREIGN_OBJECT. For this example the FOREIGN_OBJECT is named “FO.”

Real

Reals may be declared as either PARAMETERS or VARIABLES. VARIABLES are the only type that may be an output variable, so only reals may be output to SimSinter. All user defined types are actually of type real. The Variable Type just includes information about the units of the value, its default, and its minimum and maximum values. All of the following are valid declarations of real scalars or arrays:

PARAMETER

```
Alpha          AS REAL DEFAULT 0.8
FluidMass      AS Mass DEFAULT 1
AlphaArray     AS ARRAY(5) OF REAL
MassArray      AS ARRAY(5) OF Mass
```

VARIABLE

```
Alpha          AS REAL DEFAULT 0.8
FluidMass      AS Mass DEFAULT 1
AlphaArray     AS ARRAY(5) OF REAL
MassArray      AS ARRAY(5) OF Mass
```

Reals can have their values set in either the “SET,” “ASSIGN,” or “INITIAL” section, depending on if they are PARAMETERS or VARIABLES. For SimSinter to interpret a scalar integer as an input variable, it must not be in a for loop, and must be of the form:

```
Scalar Real: FO.Real__<InputName>()
```

Array Reals must be *in* a for loop, and be of the form (the “1” and the loop index are required):

```
Array Real: FO.Real1__<InputName>(<Loop Index>);
```

Example:

```

SET
  T101.Alpha                := FO.Real__AlphaFO();

ASSIGN
  T101.FlowIn := FO.Real__FlowInFO();
  FOR ii := 1 TO 5 DO
    T101.ArrayMass(ii)      := FO.Real1__Mass(ii);
  end

INITIAL
  T101.Height = FO.Real__HeightFO();

```

Integer

Integers must be declared in the “PARAMETER” section in gPROMS. They therefore cannot have a user defined variable type, and can only be input variables. Integers **cannot** be output variables. The DEFAULT shown below is *optional*.

```

PARAMETER
  SingleInt      AS INTEGER DEFAULT 11
  ArrayInt       AS ARRAY(2) OF INTEGER DEFAULT 12

```

Integers must have their values set in the “SET” section. For SimSinter to interpret a scalar integer as an input variable, it must not be in a for loop, and must be of the form:

```
Scalar Integer: FO.Integer__<InputName>()
```

Array Integers must be *in* a for loop, and be of the form (the “1” and the loop index are required):

```
Array Integer: FO.Integer1__<InputName>(<Loop Index>);
```

Example:

```

SET
  T101.SingleInt      := FO.Integer__SingleInt();
  FOR ii := 1 TO 2 DO
    T101.ArrayInt(ii) := FO.Integer1__ArrayInt(ii);
  End

```

String/Selectors

gPROMS does not have proper string variables, gPROMS uses selectors, which use strings like enumerations. These are passed through SimSinter as strings. But if an invalid value is passed as a string, gPROMS will throw an error.

Selectors must be declared in the “SELECTOR” section in gPROMS. They therefore cannot have a user defined variable type, and can only be input variables. Selectors **cannot** be output variables. The DEFAULT shown below is *optional*.

```
SELECTOR
  singleSelector AS ( apple, pear, banana )           DEFAULT apple
  arraySelector  AS ARRAY (3) OF ( red, yellow, blue ) DEFAULT red
```

Selectors must have their values set in the “INITIALSELECTOR” section. For SimSinter to interpret a single selector as an input variable, it must not be in a for loop, and must be of the form:

Single Selector: `FO.String__<InputName>()`

Array of Selectors must be *in* a for loop, and be of the form (the “1” and the loop index are required):

Array Selector: `FO.String1__<InputName>(<Loop Index>);`

Example:

```
INITIALSELECTOR
  T101.singleSelector := FO.String__singleSelector();
  FOR ii := 1 TO 3 DO
    T101.arraySelector(ii) := FO.String1__arraySelector(ii);
  END
```

Table 1: Foreign Object Method Types Reference Table

Type	Foreign Object Method
Scalar Real	<code>FO.Real__<InputName>()</code>
Array Real	<code>FO.Real1__<InputName>(<Loop Index>)</code>
Scalar Integer	<code>FO.Integer__<InputName>()</code>
Array Integer	<code>FO.Integer1__<InputName>(<Loop Index>)</code>
Scalar Selector	<code>FO.String__<InputName>()</code>
Array Selector	<code>FO.String1__<InputName>(<Loop Index>)</code>

5.3 Parenthesis at the End of Input Variable Reads in gPROMS

All the Foreign Object methods that are used to import values from SimSinter have parenthesis at the end. In the case of arrays, those parenthesis contain the loop index, but in scalars they are empty. It is easy to forget to include the parenthesis in the scalar version, so the user must be careful. gPROMS will not catch the mistake, and SimSinter will misinterpret the reference in that case. SimSinter will call the input variable something like “`Real__<name>`” which gPROMS will be unable to interpret.

5.4 SimSinter Cannot Parse Models or Variable Types from Add-On Libraries such as PML

When SinterConfigGUI configures a gPROMS simulation, it parses the .gPJ file to discover what variables are available for reading and writing. Unfortunately, types from add-on libraries such as PML and gCCS are not included in the .gPJ file by default. If the user wants to get or set a variable that is either, part-of an add on model, or has a type from an add-on variable type, the user has two options.

1. Copy the necessary model or variable type from the library into the user's project. This is only possible with open libraries such as PML. If the models and variable types are included in the .gPJ file, then SimSinter can parse them and the user can use them as input or output variables.
2. Put a connecting variables into the process. The user may define a new variable in the process that is equal to the variable in the library model, or that has a user defined type. Then *that* variable may be used as an input or output variable that just passes the variable to the actual target.

This method is most useful for use with encrypted libraries, which the user does not have access to the internals of, and SimSinter cannot parse. (e.g. gCCS)

Input Variable Tutorial:

In this case we have four input variables we wish to set in a condenser unit pulled from a the gCCS library. They are named Flowsheet.Condenser.InletCoolingWater.F, Flowsheet.Condenser.InletSteam.F, Flowsheet.Deaerator.IntletStream.F, and Flowsheet.FeedwaterHeater.InletStream.F.

SimSinter was unable to set these variables directly because they are inside the condenser unit, which is in the gCCS library. (See the comments in the ASSIGN block of Figure 39 for examples of what DIDN'T work.)

- a. First make a parameter of the correct type for each of the three variables. (See Figure 38: Creating and Setting the connecting parameters)

- b. Next set the three parameters with values from Foreign Object.

```

CondenserPart_FO123 (CondenserPart_FO123)
1 UNIT
2
3 Flowsheet AS condenser_section
4
5 PARAMETER
6     FO      as FOREIGN_OBJECT
7     CWINF AS REAL
8     CONDENSERSTEAMINF AS REAL
9     DEAERATORSTEAMINF AS REAL
10    HEATERSTEAMINF AS REAL
11
12 VARIABLE
13     coolingwateroutletf AS MassFlowrate
14     sinkutilityFWp AS Pressure
15     sinkutilityFWf AS MassFlowrate
16
17 SET
18     FO := "SimpleEventFOI::dummy";
19     CWINF := FO.Real__CWINFLOWFO();
20     CONDENSERSTEAMINF := FO.Real__CONDENSERSTEAMINFLOWFO();
21     DEAERATORSTEAMINF := FO.Real__DEAERATORSTEAMINFLOWFO();
22     HEATERSTEAMINF := FO.Real__HEATERSTEAMINFLOWFO();
23 # Start Unit Specifications
24 WITHIN Flowsheet DO

```

Figure 38: Creating and Setting the connecting parameters

- c. Finally, assign the variables in the condenser in the ASSIGN section. (Figure Figure 39: Assigning the values of the connecting parameters to the library condenser variables.)

```

CondenserPart_FO123 (CondenserPart_FO123)
127     phys_prop := "Multiflash::mass:noderiv:water.mfl
128     END # WITHIN To_Condenser
129     WITHIN To_Deaerator DO
130         phys_prop := "Multiflash::mass:noderiv:water.mfl
131     END # WITHIN To_Deaerator
132     WITHIN To_FW_heater DO
133         phys_prop := "Multiflash::mass:noderiv:water.mfl
134     END # WITHIN To_FW_heater
135     END # WITHIN Flowsheet
136 # End Unit Specifications
137
138
139 EQUATION
140     coolingwateroutletf = Flowsheet.CoolingWaterOut.F;
141     sinkutilityFWp = Flowsheet.SinkUtilityFW.p;
142     sinkutilityFWf = Flowsheet.SinkUtilityFW.F;
143
144
145 ASSIGN
146     Flowsheet.Condenser.InletCoolingWater.F := CWINF;
147     Flowsheet.Condenser.InletSteam.F := CONDENSERSTEAMINF;
148     Flowsheet.Deaerator.InletSteam.F := DEAERATORSTEAMINF;
149     Flowsheet.FeedwaterHeater.InletSteam.F := HEATERSTEAMINF;

```

Figure 39: Assigning the values of the connecting parameters to the library condenser variables.

Output Variable Tutorial:

We have three output variables we want to get from our simulation, but they are inside the gCCS library, which can't be accessed by SimSinter. The are: Flowsheet.ReheatOut.F, Flowsheet.HPSsteam.p, and Flowsheet.HPSsteam.T.

- a. First declare three variables of the correct types that will be used as the output variables. (Note, these variable types CANNOT come from the encrypted library, you may need to define your own.) See Figure 40: Declaing output connecting variables.

```

11
12 VARIABLE
13     coolingwateroutletf AS MassFlowrate
14     sinkutilityFWp AS Pressure
15     sinkutilityFWf AS MassFlowrate
16

```

Figure 40: Declaing output connecting variables

- b. Finally set the connecting variables equal to the desired library output variables in the EQUATION section. See Figure 41: Connecting the output variables.

```

139 EQUATION
140     coolingwateroutletf = Flowsheet.CoolingWaterOut.F;
141     sinkutilityFWp = Flowsheet.SinkUtilityFW.p;
142     sinkutilityFWf = Flowsheet.SinkUtilityFW.F;
143

```

Figure 41: Connecting the output variables

5.5 Solution Parameters gPLOT is REQUIRED in the Process

If a process is run from gO:Run_XML without gPLOT enabled, then all the values returned from the simulation will be '0'. In other words, to get any output variable with SimSinter, your gPROMS process MUST have gPLOT enabled in the SOLUTIONPARAMETERS section. See Figure 42.

```

334 SOLUTIONPARAMETERS
335     gPLOT := ON
336     InitialisationProcedure := OFF
337

```

Figure 42: SOLUTIONPARAMETERS gPLOT := ON is required

5.6 SimSinter Cannot use Multi-Dimensional Arrays as Inputs or Outputs

gPROMS supports arrays of arbitrary dimension. Unfortunately SimSinter only supports single-dimensional vectors. So SinterConfigGUI will simply ignore multidimensional array variables. If the user wishes to input or output a multidimensional array, a connecting 1D vector will have to be used, as in 3.5.2

5.7 Variable and Parameter Defaults Defined in gPROMS

When a variable or parameter is declared in gPROMS, it may be declared with a default. Also, variable types often include a default. SinterConfigGUI does its best to read those defaults and import them into the SinterConfigGUI as input variable defaults. However, gPROMS allows default to be defined in reference to other variables or functions. SimSinter cannot interpret variable or function values, so those defaults are skipped, and set to "0." So:

- **X** pi AS REAL DEFAULT 2*ACOS(0)
- **O** pi AS REAL DEFAULT 3.1415926

5.8 gO:Run_XML License Required

SimSinter runs gPROMS simulations with a tool that is installed with ModelBuilder named "gO:Run_XML." However, having a license for ModelBuilder does imply a license for gO:Run_XML is also available. In ModelBuilder 4.0.0 gO:Run_XML requires both gSIM_7 and gSRE_7 licenses, but as of 4.1.0, that has changed. Please confirm with your gPROMS sales representative that you have the correct licenses to run gO:Run_XML.

5.9 Simulations are Configured with .gPJ Files, but Run with .gENCRYPT

SimSinter requires two different representations of a gPROMS simulation for the two different phases SimSinter goes through.

1. The configuration phase of SimSinter, performed via SinterConfigGUI, requires the .gPJ file. The .gPJ file is not encrypted, so SimSinter, or anyone else, can read it and discover things about the model. If the model is secret, **do not** distribute the .gPJ file. The .gPJ file is only required for simulation configuration, so if the model is secret, the user should perform the SimSinter configuration themselves.
2. The run phase of SimSinter, performed via Turbine or ConsoleSinter, requires a .gENCRYPT file, exported from the project (**not** the process). This is because the PSE tool, gO:Run_XML, requires an encrypted file so that developers can distribute secret models safely to users. SimSinter cannot run a simulation from a .gPJ file.

5.10 The Name of the gENCRYPT File is Based on the Project File Name

When exporting a .gENCRYPT file, ModelBuilder will automatically give the .gENCRYPT file the name “<Project Name>.gENCRYPT,” just as the .gPJ file is named “<Project Name>.gPJ.” It is recommended that the user does not change the name of the .gENCRYPT file. If the user changes the file name, the user will have to edit the SinterConfig.json file as well to update it, as there is no way to change the name in SinterConfigGUI.

If the user decides to change the name of either the .gPJ or .gENCRYPT files, those entries may be found in the SinterConfigFile under “model” for the .gENCRYPT file, and “simulationDescriptionFile” for the .gPJ file.

```
"model": "<ProjectName>.gENCRYPT",  
"simulationDescriptionFile": "<ProjectName>.gPJ",
```

6.0 DEBUGGING

6.1 How to Debug by Yourself

Most issues with running gPROMS under SimSinter are related to issues with gO:Run_XML. So it is often helpful to run gO:Run_XML by itself, without SimSinter. This often provides some useful output the user otherwise wouldn't see from SimSinter.

1. To run gO:Run_XML, open a windows command prompt by opening the start menu, and typing “cmd”, and hitting ‘enter.’

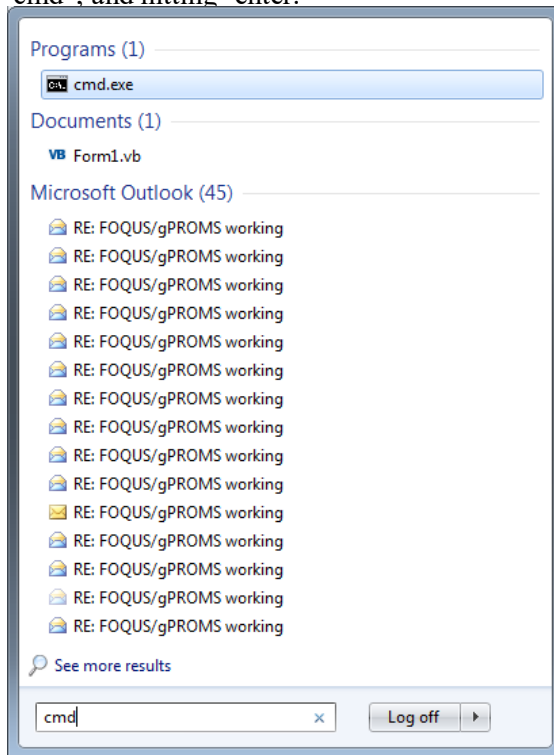


Figure 43: Launching a command prompt

2. In the command prompt, change directory to your simulation. Type “cd <directory name>” and hit ‘enter.’

In this case, we will use the demonstration simulation installed by SimSinter in
c:\SimSinterFiles\gPROMS_Test

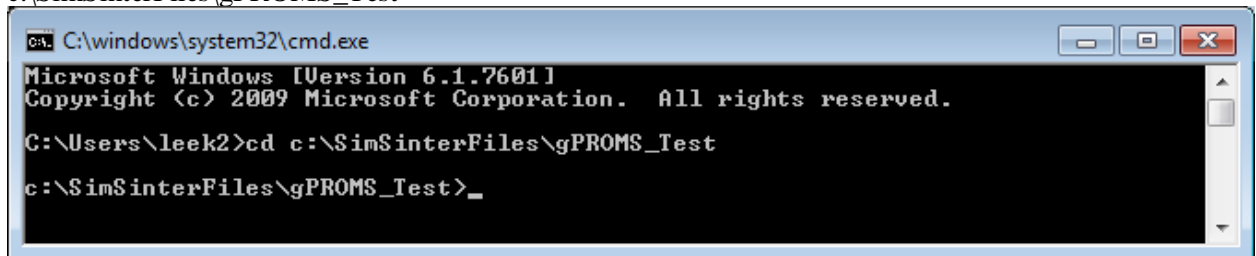


Figure 44: Change Directory to the simulation directory

3. Now type ‘dir’ and press enter. This will list the files in the directory. If you have run sinter on this simulation before, even if it failed, there should be a sinterInput.xml file. That is the input

file to gO:Run_XML.

```

C:\windows\system32\cmd.exe
Volume in drive C is OSDisk
Volume Serial Number is FA9C-5C8A

Directory of c:\SimSinterFiles\gPROMS_Test

04/04/2016  11:16 AM    <DIR>          .
04/04/2016  11:16 AM    <DIR>          ..
03/17/2016  09:44 AM             4,716 BufferTank_F0.gENCRYPT
03/17/2016  09:44 AM            15,457 BufferTank_F0.gPJ
03/24/2016  02:37 PM             5,026 BufferTank_F0.json
04/01/2016  10:43 AM             188 BufferTank_F0._time.txt
03/17/2016  09:59 AM             511 BufferTank_inputs.json
03/31/2016  03:09 PM    <DIR>          code
04/01/2016  10:43 AM            2,169 goo.out
04/01/2016  10:43 AM    <DIR>          input
03/31/2016  04:21 PM            2,169 out.json
04/04/2016  11:16 AM             463 out.xml
04/01/2016  10:43 AM    <DIR>          output
04/01/2016  10:43 AM             206 runtimings.txt
03/31/2016  03:09 PM    <DIR>          sinter
04/01/2016  10:43 AM            1,989 sinterInput.xml
04/01/2016  10:43 AM            1,005 sinteroutput.xml
               11 File(s)            33,899 bytes
               6 Dir(s)  335,322,652,672 bytes free

c:\SimSinterFiles\gPROMS_Test>

```

Figure 45: Checking for the sinterInput.xml file

4. If the sinterInput.xml file is there, then we can try running gO:Run_XML on it. There are three possible methods for running it:
 - a. The simplest method is to allow windows to find it itself via the PATH variable. However, this relies on the user have added gPROMS to the path at installation time, and multiple versions of gPROMS being installed on the machine may make it difficult to figure out which one is actually being run. But this is the command:


```
gO:Run_XML.exe sinterInput.xml out.xml
```
 - b. SimSinter uses the GPROMSHOME environment variable to locate gO:Run_XML, so if you want to be sure to run that same version as SimSinter, use this command (include the quotes):


```
"%GPROMSHOME%\bin\gO:Run_XML" sinterInput.xml out.xml
```
 - c. If you want to run a particular version of gO:Run_XML, you will have to specify the whole path. Which will be something like this (include the quotes):


```
"C:\Program Files\PSE\gPROMS-core_4.2.0.54965\bin\gO:Run_XML" sinterInput.xml out.xml
```
5. After running gO:Run_XML, you should have some useful output that will allow you to debug the error. Please see the next section for more details.

6.2 Known Issues

6.2.1 License issue, sim doesn't run

By far the most common issues we have seen with running gPROMS have been licensing issues. This is because the ModelBuilder license and the gO:Run_XML license are different licenses, so just because you have the ModelBuilder license, doesn't mean you can run gO:Run_XML. To add to the confusion, as of ModelBuilder 4.1, PSE has added a new licensing scheme, so either of two licenses will allow the

user to run gO:Run_XML: gSRE_7, or 9230_GPROMS_ENCRYPTED. gSRE_7 is the old license type, and 9230_GPROMS_ENCRYPTED is the new one.

This text indicates that gO:Run_XML could not find a valid license:

```

C:\windows\system32\cmd.exe
c:\SimSinterFiles\gPROMS_Test>gORUN_xml sinterInput.xml out.xml
Requesting 9230_GPROMS_ENCRYPTED license from server.
Requesting gSRE_7 license from server.
Disconnected from license server

gPROMS <R> - Version 4.2.0 <x64> for Microsoft Windows 7 Service Pack 1 Nov 30 2015
general PROcess Modelling System

Copyright (c) 1997-2015 Process Systems Enterprise Ltd

gPROMS is a trademark of Process Systems Enterprise Ltd.
All other registered and pending trademarks mentioned in this
software are considered the sole property of their respective
owners. All rights reserved.

This computer program is protected by copyright law and
international treaties. Unauthorised copying, reproduction or
distribution of this program is a violation of applicable laws.

Visit WWW at http://www.psenterprise.com/ for more information.
E-mail support.gPROMS@psenterprise.com for product support.

ERROR! Failed to start gPROMS; the function gproms_start() has not been called,
or a license is not available. <status = 0>

c:\SimSinterFiles\gPROMS_Test>

```

Figure 46: No valid gO:Run_XML license

6.2.2 ERROR: “gPROMS executable gO:Run_XML.exe could not be found”

This error should be rare, and only occur if something has gone wrong with gPROMS installation. SimSinter looks for gO:Run_XML both in %GPROMSHOME%\bin, and in the %PATH% environment variable. This error only appears if gO:Run_XML can’t be found in either.

In that case, please ensure gPROMS 4.0.0 is installed.

If so, open a Windows Command line and type “echo %GPROMSHOME%” make sure it looks reasonable.

If so, please contact ccsi-support for more help.

6.2.3 goORUN_xml produces “Unable to obtain license from server” but runs the simulation

With version of gPROMS 4.1 or newer, when running gO:Run_XML, it may complain about “Unable to obtain license from server,” but then run the simulation anyway. This is due to the new licensing scheme adopted as of version 4.1.0. It is not actually a problem. If the simulation runs, you have a license, but if you don’t have the NEW style of license, gO:Run_XML outputs a lot of useless warnings, as seen below. Just ignore it.

Example text of license confusion:

```

Requesting 9230_GPROMS_ENCRYPTED license from server.
Unable to obtain license from server.
Failed to get licence: License server system does not support this feature.

```


Feature: 9230_GPROMS_ENCRYPTED
 License path: @flex1.acceleratecarboncapture.org;C:\Program Files\PSE\gPROMS-core_4.1.0.54941\licenses*.lic;license.dat;*.lic;
 Trimmed for space
 Requesting gSRE_7 license from server.
 License granted by server(s) flex1.acceleratecarboncapture.org.
 Trimmed for space

Requesting 9230_SIM license from server.
 Unable to obtain license from server.
 Failed to get licence: License server system does not support this feature.
 Feature: 9230_SIM
 Trimmed for space
 Requesting gSIM_7 license from server.
 License granted by server(s) flex1.acceleratecarboncapture.org.
 Loaded "gPLOT.dll".
 Execution of SimulateTank_sinter completed successfully.

Simulation took 0 seconds.
 Total CPU time: 0.140s (56% system time)

Returning gSIM_7 license to server.
 License returned to server.
 Returning gSRE_7 license to server.
 License returned to server.
 Disconnected from license server

6.2.4 The Simulation seems to have Succeeded, but all the Output Variables are '0'

This can be difficult to debug because gO:Run_XML does not throw any errors if non-existent output variables are requested, it just returns '0' for them. So there are a couple of possibilities:

1. Check that the .gENCRYPT file and .gPJ file you built the Sinter Configuration from match. It's easy to forget to generate a new .gENCRYPT after updating the .gPJ.
2. Check that the output variable names and paths are correct in sinterInput.xml. In the Sinter Configuration file, the output variable path will start with the process name (e.g. processname.unit.variablename), but in sinterInput.xml the report variable will NOT start with the processname. (e.g unit.variablename.)
3. Check that your gPROMS process includes gPLOT := ON in the SOLUTIONPARAMETERS section. See Figure 42: SOLUTIONPARAMETERS
 gPLOT := ON is required

6.3 How to get help

No software is perfect, and while SimSinter is fairly well tested, some issues are expected.

If a developer would like to add to SimSinter, or fix bugs in the software, the source code can be provided. Request the code by sending an e-mail to ccsi-support@acceleratecarboncapture.org.

If a user encounters a bug, send a detailed description of the bug, along with a reproducer (if possible) to ccsi-support@acceleratecarboncapture.org.

6.4 Reporting Issues

Report all issues, feature requests, and help requests by sending an e-mail to ccsi-support@acceleratecarboncapture.org.