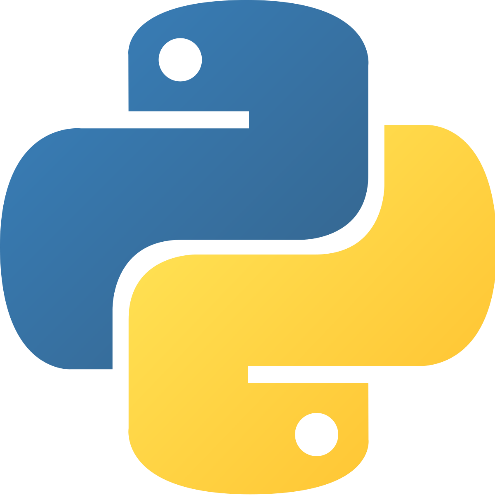
Documentation of

Aspen Plus-Python interface

Python library for automating the Aspen flowsheet, blackbox solution to input, run, and receive data from Aspen.

Receive results

Input design choices

Table of Contents

[1. Introduction: 6](#_Toc107409612)

[1.1. How to use the documentation 6](#_Toc107409613)

[1.2. File Formats in Aspen Plus 7](#_Toc107409614)

[1.3. Variable manager 8](#_Toc107409615)

[1.4. Etymology of function names: 10](#_Toc107409616)

[2. Common use cases: 12](#_Toc107409617)

[2.1. Tutorial: Learn to connect, run, get reports, set variables, 12](#_Toc107409618)

[2.2. Case 1.1: Single variable varying: 15](#_Toc107409619)

[2.3. Case 1.2. Double variable varying 16](#_Toc107409620)

[2.4. Scipy Optimization library: 18](#_Toc107409621)

[2.4.1. Case 2.1. Genetic algorithm 20](#_Toc107409622)

[2.4.2. Case 2.2. Brute force algorithm: 21](#_Toc107409623)

[2.5. Future improvements to the library: 23](#_Toc107409624)

[3. Power functions 25](#_Toc107409625)

[3.1. Flow sheeting, Placing, Removing 25](#_Toc107409626)

[3.2. Running Simulations: 25](#_Toc107409627)

[3.3. Saving Reports, Exporting Files: 26](#_Toc107409628)

[4. Higher level functions: 26](#_Toc107409629)

[5. RADFRAC Inputs 28](#_Toc107409630)

[5.1. Configuration: Page 1 28](#_Toc107409631)

[5.2. Streams Page 2 29](#_Toc107409632)

[5.3. Pressures Page 3 30](#_Toc107409633)

[5.4. Condenser Page 4 32](#_Toc107409634)

[5.5. Reboiler Page 5 34](#_Toc107409635)

[6. RADFRAC OUTPUTS 35](#_Toc107409636)

[6.1. Summary: Page 1 Output 35](#_Toc107409637)

[6.2. Balance: Page 2 Output 36](#_Toc107409638)

[6.3. Split fractions: Page 3 Output 37](#_Toc107409639)

[6.4. Reboiler: Page 4 Output 37](#_Toc107409640)

[6.5. Utilities: Page 5 Output 39](#_Toc107409641)

[6.6. Stage Utilities: Page 6 Output 40](#_Toc107409642)

[6.7. Status: Page 7 Output 40](#_Toc107409643)

[7. Heater Input 41](#_Toc107409644)

[7.1. Specifications: Input Page 1 41](#_Toc107409645)

[7.2. Flash Options Input Page 2: 42](#_Toc107409646)

[8. Heater Output 43](#_Toc107409647)

[8.1. Summary Page 1 43](#_Toc107409648)

[8.2. Balance Page 2 43](#_Toc107409649)

[8.3. Phase equilibrium Page 3 44](#_Toc107409650)

[8.4. Utilities Page 4 45](#_Toc107409651)

[8.5. Status Page 5 45](#_Toc107409652)

[9. RPLUG Input 46](#_Toc107409653)

[9.1. Reactor type Input Page 1: 46](#_Toc107409654)

[9.2. General Reactor Configurations Input Page 2: 50](#_Toc107409655)

[9.3. Streams: Input Page 3: 51](#_Toc107409656)

[9.4. Reactions: Input Page 4: 51](#_Toc107409657)

[9.5. Pressure specification Input Page 5: 52](#_Toc107409658)

[9.6. Reactor holdup Input Page 6: 53](#_Toc107409659)

[9.7. Catalysts Input Page 7: 54](#_Toc107409660)

[10. RPLUG OUTPUT 55](#_Toc107409661)

[10.1. Summary: Page 1 55](#_Toc107409662)

[10.2. Balance: Page 2 56](#_Toc107409663)

[10.3. Distribution: Page 3 57](#_Toc107409664)

[10.4. Polymer Attributes: Page 4 57](#_Toc107409665)

[10.5. Status: Page 5 57](#_Toc107409666)

[11. RCSTR Inputs 58](#_Toc107409667)

[11.1. Specifications Page 1: 58](#_Toc107409668)

[11.2. Streams Page 2: 60](#_Toc107409669)

[11.3. Kinetics Page 3: 61](#_Toc107409670)

[11.4. Particle Size Determination PSD Page 4: 62](#_Toc107409671)

[11.5. Component Attributes Page 5 63](#_Toc107409672)

[11.6. Utilities Page 6: 63](#_Toc107409673)

[11.7. Catalyst Page 7: 63](#_Toc107409674)

[12. RCSTR Output: 64](#_Toc107409675)

[12.1. Summary Page 1: 64](#_Toc107409676)

[12.2. Balance Page 2: 65](#_Toc107409677)

[12.3. Reaction Kinetics Page 3: 66](#_Toc107409678)

[12.4. Component Generation Rates Page 4: 66](#_Toc107409679)

[12.5. Custom Reaction Variables Page 5: 66](#_Toc107409680)

[12.6. Utility Usage Page 6: 66](#_Toc107409681)

[12.7. Distributions Page 7: 67](#_Toc107409682)

[12.8. Polymer Attributes Page 8: 67](#_Toc107409683)

[12.9. Crystallization Page 9: 67](#_Toc107409684)

[12.10. Status Page 10: 67](#_Toc107409685)

[13. Mixer Input 68](#_Toc107409686)

[13.1. Flash Option Page 1: 68](#_Toc107409687)

[14. MIXER Output 69](#_Toc107409688)

[14.1. Summary Page 1: 69](#_Toc107409689)

[14.2. Balance Page 2: 69](#_Toc107409690)

[14.3. Status Page 3: 70](#_Toc107409691)

[15. RYIELD Input 71](#_Toc107409692)

[15.1. Specifications Page 1 71](#_Toc107409693)

[15.2. Streams Page 2: 72](#_Toc107409694)

[15.3. Yield Page 3: 72](#_Toc107409695)

[15.4. Flash Options Page 4: 74](#_Toc107409696)

[15.5. Particle Size Determination PSD Page 5: 74](#_Toc107409697)

[15.6. Component Attribute Page 6 75](#_Toc107409698)

[15.7. Component mapping Page 7 75](#_Toc107409699)

[16. RYIELD Output: 76](#_Toc107409700)

[16.1. Summary Page 1: 76](#_Toc107409701)

[16.2. Balance Page 2: 76](#_Toc107409702)

[16.3. Phase Equilibrium Page 3: 77](#_Toc107409703)

[16.4. Weight distribution Page 4: 78](#_Toc107409704)

[16.5. Pseudocomp Breakdown Page 5: 78](#_Toc107409705)

[16.6. Utility usage Page 6: 78](#_Toc107409706)

[16.7. Status Page 7: 78](#_Toc107409707)

[17. FLASH2 Input 79](#_Toc107409708)

[17.1. Specifications Page 1: 79](#_Toc107409709)

[17.2. Flash Option Page 2: 80](#_Toc107409710)

[17.3. Entrainment Page 3: 81](#_Toc107409711)

[17.4. Particle Size Determination (PSD) Page 4: 81](#_Toc107409712)

[17.5. Utilities Page 5: 82](#_Toc107409713)

[18. FLASH2 Output: 83](#_Toc107409714)

[18.1. Summary PAGE 1 83](#_Toc107409715)

[18.2. Balance PAGE 2: 83](#_Toc107409716)

[18.3. Phase Equilibrium PAGE 3: 84](#_Toc107409717)

[18.4. Utility Usage PAGE 4: 85](#_Toc107409718)

[18.5. Status PAGE 5: 85](#_Toc107409719)

[19. DSTWU Input 86](#_Toc107409720)

[19.1. Specification PAGE 1: 86](#_Toc107409721)

[19.2. Calculation Options PAGE 2: 87](#_Toc107409722)

[19.3. Convergence PAGE 3: 88](#_Toc107409723)

[20. DSTWU Outputs: 88](#_Toc107409724)

[20.1. Summary PAGE 1: 89](#_Toc107409725)

[20.2. Balance PAGE 2: 89](#_Toc107409726)

[20.3. Reflux ratio profile PAGE 3: 89](#_Toc107409727)

[21.4. Status PAGE 4: 89](#_Toc107409728)

[21. FSPLITTER Input 89](#_Toc107409729)

[21.1. Specification PAGE 1: 90](#_Toc107409730)

[21.2. Flash Options PAGE 2: 90](#_Toc107409731)

[21.3. Key Components PAGE 3 90](#_Toc107409732)

[22. FSPLITTER Output 90](#_Toc107409733)

[22.1. Summary PAGE 1: 90](#_Toc107409734)

[22.2. Balances PAGE 2: 90](#_Toc107409735)

[22.3. Status PAGE 3: 90](#_Toc107409736)

[23. STREAM INPUT 90](#_Toc107409737)

[23.1. Specifications PAGE 1 90](#_Toc107409738)

[23.2. CI Solid PAGE 2 90](#_Toc107409739)

[23.3. NC Solid PAGE 3 90](#_Toc107409740)

[23.4. Flash Options PAGE 4 90](#_Toc107409741)

[23.5. EO Options PAGE 5 90](#_Toc107409742)

[23.6. Costing PAGE 6 90](#_Toc107409743)

[24. STREAM OUTPUTS: 90](#_Toc107409744)

# Introduction:

The following report represents the documentation for a Python library that has been created to automate Aspen Plus from an interactive user input, toward a fully automated black box solution. The Python library has functions for setting the inputs and retrieving the outputs of the most important process equipment’s, as well as creating/altering the flowsheet. Using this, most capabilities of the Aspen Plus user interface are transferred into python. It is possible to control Aspen without ever opening it and setting manual inputs in Aspen. This is of course still incomplete since Aspen Plus is a very extensive program.

**Motivation:**

The initial motivation for generating this library was the usage of Reinforcement learning techniques in chemical engineering to solve problems such as: the sequencing of distillation columns. This was done in a Bachelor thesis upon which this is based [(Bachelorthesis).](https://github.com/YouMayCallMeJesus/ReinforcementlearningWithDestillationColumns/blob/main/Notes%20for%20the%20Bachelor%20assignment%2027%20long.docx) This proof of concept was successful in implementing the system but since it was the first python code there were considerable problems with both the AI as well as the Aspen Python API side of the thesis. For further development, it was decided that they are to be split apart into separate projects. For more details about the development of the Artificial intelligence side of it please read: [(Migleys AI environment).](https://github.com/lollcat/Aspen-RL)

While creating the Aspen Python interface (API) it was discovered that there is a multitude of other applications where it would be advantageous to make use of automated optimization. Examples of this are: placing new blocks, optimizing the entire flowsheet with newer optimization algorithms, dynamic error handling for convergence errors, complete mapping of a variable space, and much more. For more case examples and practical applications go to chapter 2.

## How to use the documentation

As you can see in the number of pages for this documentation, there are too many details in this library that you most likely do not care about. To minimize your labor, I would advise the following procedure to get started with the library:

1. Go through the tutorial code and documentation and test out the cases

* Helps you connect your Python/Aspen and get a first taste of how it works

1. Read the Power function section for flow sheeting, saving, placing, debugging etc,

* Foundational functions which are needed for any program to run

1. Read the input section of the equipment which you want to use
   1. Make choices for how you want to simulate it
   2. Check out the basic build tutorial
2. Read the output section of your equipment
   1. Decide on what you find important in the outputs
3. Start running simulations and change some parameters
4. Build your own functions to optimize the process.

* For this check out: Chapter 1.3.

If you have any trouble, email me: Richardxtenxhagen@gmail.com

## File Formats in Aspen Plus

Aspen Plus has many different methods for outputting data to the User. These can be divided into methods that are meant to be read by humans and ones which are to be used by the software. The ones which can be read by humans are extremely useful for debugging and just “seeing” what is happening. On the other hand, if you are trying to connect Aspen into some optimization algorithm you want the data to be in a variable form which you can set and read. This fundamental dualism represents a major problem that can be partially solved using an API or by using some exported files. These are very useful since it allows for the automatic generation of backups and reports for the work.

Here is the listing of the currently supported Aspen Plus files:

.apw: Aspen Plus document

This is an intermediate representation of the flowsheet in front of you. After loading, it starts exactly where you left off with all the results, runtimes, and everything else you are currently seeing.

.bkp: Backup Files

This is a compact version of apw. It contains all the inputs but no results. Needs to run again to generate the results. It has the advantages of being smaller file size, portable, version independent, can be imported as a section into other sheets, and is thereby more useful when you mean to load some part of the program. It can be read by a human (as a txt but it is painful)

.apt: template file

Default settings for some part of your flowsheet can be used for: units, property sets, stream report formats, and property option sets.

.inp: Input files

A compact summary which can be read out as a text file. It is used for standalone Aspen Plus engine runs. It is the File which Aspen uses for input but at the same time it also represents readable documentation for the human. It seems to be an “old school method” of setting the inputs before there was the interactive Aspen Plus GUI. This can be edited automatically and with some string editing skills and this can also be used to edit the inputs. The format of the txt file is made for the human eye and can be read like a report.

.rep: Report File:

These are reports on specific simulation results. They can be used as documentation as they are readable as text files. They represent the Ideal method of receiving output data from the Aspen system. It is also possible to generate your own templates for reports if you want to monitor something special.

.sum: Summary Files:

These are summaries of all data inside of the interface. The format is ASCII files. This is the method that other programs can use to import the results of your simulation into them. They can also be loaded back into the Aspen Plus User Interface.

.cpm: Run Messages Files:

These Files include the Error Warnings and diagnostics for each run. The number of Messages can be controlled with the Setup Specifications Diagnostics sheet. Or on the BlockOptions diagnostics. If they are not exported from the simulation, they are deleted automatically.

.his: History File:

This file contains the input summary, errors, warnings, and diagnostic messages of a single run.

These file formats can play an important role in creating, changing, and loading different Aspen Plus models. It will be possible to export the files for each type and these will be very useful for the making of automatic comparisons of different results of automation

## Variable manager

This entire automation library was made possible by the Aspen Plus ActiveX automation Server which is detailed in the Aspen plus User manual V10. If one wants to edit the library or change things it is advised to read this since it gives an overview of the methods (although the code, there is written in VBA). The basis of it all is the variable tree, which can be seen below in Figure 1. It is structured similarly to the Windows File explorer where there are many folders and subfolders. The Variable manager can be found at: Customize/Variable Explorer. Please look at it in your Aspen file since this will become your new best friend and you will use it constantly to debug the library.

Graphical user interface, application, Word

Description automatically generated

Figure Variable manager

In the Aspen Plus language, each of these points is called a node. Each node has a list of properties that are assigned to it.

Each node can also have a “child” aka a subnode aka a folder in your folder. In this report the dot notation will be used to show the subnode structure: this means that the top node is named first and the subsequent children are then followed by a point. An example of such a path can be seen below which is also used in Figure 1:

Root.Data.BLOCKS.B1.Input.UnitSet

There are far too many variables to go through and figure out what each one of them does. To mitigate this, Aspen has given us the capability to find the variable path by first right-click copying the variable in the interactive interface, and then secondly searching for the path in the variable manager. This allows for the determination of each variable inside of each block. If you increase the size and power of the current library, please send me an email such that I can also include your functions.

Graphical user interface, application, Word

Description automatically generatedGraphical user interface, application, Word

Description automatically generated

Figure Copying variable interactively Figure Finding copied variable path

This process of finding each path has been completed for most of the inputs and outputs of the streams and the following equipment: DSTWU, Flash2, Radfrac, Mixer, Heater, RYield, RPlug, and RCSTR. Finding all these pathways has been the main workload for this project because of the manual labor needed for doing it completely. Starting in chapter 3, there are tables that contain all the different pathways which have been found and the function names which are needed to call them. These variables are not very useful on their own since we are still not able to edit or export them. For this, we make use of the “Win32.com” Python - Windows API for being able to access the Aspen Plus variables and edit them. With this library imported it is possible to read and edit the nodes just as if they were files in Windows File Explorer.

Example for the nomenclature of editing or reading the node value:

sim.BLK\_RADFRAC\_Set\_CondenserDuty(1000)

**Common nodes which are used:**

Now that we know how to edit the files, we can look at some of the most used nodes which we have in the python library:

AspenSimulation equivalent to win32.gencache.EnsureDispatch("Apwn.Document")

The top layer of the node system in Aspen, it represents the entire Aspen program you are currently dealing with. It is used when changing settings or when searching for lower layers.

Interestingly you can make multiple instantiations of this for a single Aspen Plus file at the same time. This allows for the multithreading and running of multiple versions of your code on a single Aspen. In practice, this would be done by calling the instantiations of the “sim” multiple times. (This is relevant if you want to do some multi-threading)

Below this top layer, it is useful to make some artificial node names to simplify the path names of each variable. These can be called as a variable in python and replace the long path string. The two used here are:

STRM equivalent to sim.AspenSimulation.Data.Streams

This node contains the list of all the different “Streamnames” nodes which contain all the data about the specific stream in the flowsheet.

STRM.Streamname.Output Subnode of stream containing the output data

STRM.Streamname.Input Subnode of stream containing the input data

BLK: equivalent to sim.AspenSimulation.Data.Blocks

This node is used as a shortcut to the data which contains all the different process equipment blocks. It contains just like the stream a list of all the equipment and an output and input node which gives the data we seek.

**Node Attributes:**

Each node can have various characteristics which are called attributes, these are used to describe the point in the network. The most common one is node. The value which contains the numerical value of the node, for example for the reflux ratio: “BLK.Input.RR.Value = 1.5”

The units which are used for the value can be found by using the .Unitrow and .Unitcol attribute. These can be used in combination with the table of units which contains all the possible units. The most common value for both is 0. This also depends on the .Basis which is used for the calculations and it can be either MASS or MOLE.

If the value is not an integer but one of a choice of possibilities then the .Optionlist is useful since it can tell you all possible inputs that are allowed.

## Etymology of function names:

**General Powerfunction**

General power functions are used to interact with the sheet. These range from functions like “Run simulation” and “DeleteAllBlocks” towards “SaveAspen”. These are very useful independent of the exact flow diagram which will be created and they do not follow any special naming convention. I would advise reading chapter 3 since they are important.

**Notation difference between GET and SET**

When the API interacts with Aspen there are two distinct ways of communicating and that is to Get and Set variables. If the function name contains “GET” it will retrieve a value from Aspen and “SET” will define a variable in the flowsheet.

**Block functions:**

The second category is the “Block input functions”. These are always specific to singular process equipments and there are limits on which equipment has been prepared for this. Currently, it is possible to use: DSTWU, Flash2, RADFRAC, Mixer, Splitter, Heater, RYIELD, RPLUG, and RCSTR. In the future, there might be further developments on this to include even more equipment. The Block input functions all start with the prefix: “BLK\_” followed by the name of the equipment for example: “BLK\_DSTWU\_Set\_” and then the input variable is named. Here is an example:

CondTemp = BLK\_RADFRAC\_Get\_Condenser\_Temperature()

As a word of warning: Sometimes a previous input choice (aka an “Option”) enabled the usage of more inputs that were previously not possible. Below is an example of this:

BLK\_RADFRAC\_Set\_PARTIAL\_V\_L\_CondenserOption(“TEMP”)

This means that the CondenserOption is the actual variable that is being set and the variable name contains the dependencies. In this case, a previous choice was made for the condenser type to be “PARTIAL\_V\_L” which then enabled the choice of the CondenserOption.

These single variable functions enable the editing of the in/outputs which is useful if repeated simulations are wanted with a few varying parameters. In other cases, it is necessary to make use of higher-level functions which are composites of these lower ones. Examples can be found in the higher functions below:

**Stream input functions:**

The third category is the “Stream input functions” which are very similar to the block functions, but these make use of the STRM block as the first node.

STRM\_Set\_Temperature(200)

**Higher-level functions:**

Higher-level functions contain a large collection of functions and bundle them together to simplify the process of interacting with Aspen. These functions either intake or output dictionaries which contain all the data, and those can then be printed. They are easily found because their names are in all caps:

STRM\_GET\_OUTPUTS(“streamname”)

BLK\_DSTWU\_SET\_ALL\_INPUTS(dictionary)

Dictionary = BLK\_DSTWU\_GET\_ME\_ALL\_INPUTS\_BACK(“columnname”)

Dictionary = BLK\_DSTWU\_GET\_OUTPUTS(“columnname”)

The complete list of all the higher-level functions can be found in: chapter 4

The method by which these higher functions have been built is very crude and tends to break since they require custom error handling. This is caused by Aspen having a problem with retrieving the value of the variable which does not exist. As soon as you find this just wrap the part that gives you trouble in a try: … except: Exemption…pass. An example of this can be seen below for DSTWU:



Figure Wrapping higher-level functions in try/except

This will be necessary since the variables which result from options are only created if the option is enabled. In the case of Figure 4, one can see that the GenerateTableOption is NO which means that the GenerateTable\_FirstStage does not exist. When we now try to set the variable into this path that does not exist and this results in an error.

# Common use cases:

## Tutorial: Learn to connect, run, get reports, set variables,

The Python file for this tutorial is called TutorialCode.py

The initial tutorial is implemented in Python, and it is used to connect Aspen to Python, determine the directory path, and to go through some of the most important features of this library. This tutorial makes use of printing messages as well as requesting input from the user. These inputs are mainly used to slow down the program and allow the reading of the messages or to look at the flowsheet after alterations were made to it.

Below you can see the figure which shows the flowsheet:

Diagram

Description automatically generated

Figure Flowsheet layout for tutorial

The first step of the tutorial is instantiating the simulation class. This will be used to access the functions from the library. To be able to do this instantiating there are three inputs needed: The name of the Aspen file which will be worked on, the directory where this file can be found, and an option on whether the simulation should also open the Aspen program interactively. Generally speaking, it should be “True” for debugging and “False” when a large number of simulations are to be run since activating the visibility increases runtime, results in more errors and is troublesome if clicked while running.

sim = Simulation(AspenFileName= "Simple Problem.bkp", WorkingDirectoryPath=r"C:\Users\Richard\Desktop\AspenPythonInterface" ,VISIBILITY=True)

The second code block is the usage of a Power function to determine all the different inputs which were used in the already given Aspen tutorial file.

CISTRInputDictionary = sim.BLK\_CISTR\_GET\_ME\_ALL\_INPUTS\_BACK("B1")

RPLUGInputDictionary = sim.BLK\_RPLUG\_GET\_ME\_ALL\_INPUTS\_BACK("B3")

The functions will return a dictionary that contains all the different values with the corresponding names. These will then also be printed such that the user can look at the different options and what had been chosen.

Now that we know what selections were made for these two blocks, it is time to run the simulation. This can be done by either the sim.EngineRun or the sim.Run(), in this case we will make use of the EngineRun since we know that there will be no problems. Usually, it is advisable to use the sim.Run() since it has an internal error handling, which will rerun the simulation in case of Aspen simulation errors.

sim.EngineRun()

In the case of the tutorial, one will notice that once the simulation ran a popup message appears in Aspen. This dialog box will inform you that the simulation ran successfully and that there were no problems. To suppress further messages from appearing it is possible to make use of the following function:

sim.DialogSuppression(TrueOrFalse= True)

After having run the simulation and finding that it has run successfully it is now time to retrieve the data about the two reactors.

RPLUGOutputDictionary = sim.BLK\_RPLUG\_GET\_OUTPUTS("B3")

RCSTROutputDictionary = sim.BLK\_RCSTR\_GET\_OUTPUTS("B1")

The dictionaries which were found are then also printed which allows for the user to easily comprehend the results.

Afterward the same will also be done for the Productstreams and the result will be printed.

RPLUGStreamDictionary = sim.STRM\_GET\_OUTPUTS("RPLUGOUT")

These higher-level functions are not used that often since they are less reliable and much slower which means that single-value functions are more common. To demonstrate the functionality of those we will extract the heat duty of the CISTR and then also input a new light key recovery.

HeatDuty = sim.BLK\_RCSTR\_Get\_HeatDuty(Blockname="RCSTR")

sim.BLK\_DSTWU\_Set\_LightkeyRecovery("B1", 0.06)

The lightkey was chosen such that it would not converge since this gives us the chance to discover how convergence errors can be handled.

convergence = sim.Run()

In this case, we will be using the sim.Run() command since it will attempt to run the simulation and once it fails it will try again. In the end, a variable is returned that tells you if there was a convergence or not. This means that you can find out when there was no convergence and deal with it on a higher level of your code.

The next step would be the creation of a new distillation column after the reactor. This means that we will first place the DSTWU:

sim.BlockPlace("B6","DSTWU")

The first variable is the name of the new block and the second one describes the type. All possible types are: "RCSTR", "RPlug", "DSTWU", "Flash2", "Mixer", "Heater", "Radfrac", "Splitter", "RYield"

The next step is the placing of the streams where the Streamtyp can be “MATERIAL”, “HEAT” or “” as well as the connecting of the streams to the block:

sim.StreamPlace(Streamname="S1", Streamtype= "MATERIAL")

sim.StreamConnect(Blockname="B6",Streamname="RPLUGOUT", Portname="F(IN)")

When connecting streams to a block it is necessary to also give a portname that informs the Aspen about the position for the connection. These are predefined and if you want to see all the different ports you need to go to the variable manager.

The next step to get the distillation column to run is to set the input variables. Since I am lazy and it would be boring to go through the list and set each value on their own, we will make use of another higher-level function. In this case, we will first extract the input variables of an already existing column and then put those into the new column.

OldDSTWUinputDictionary = sim.BLK\_DSTWU\_GET\_ME\_ALL\_INPUTS\_BACK("B1")

sim.BLK\_DSTWU\_SET\_ALL\_INPUTS("B6", OldDSTWUinputDictionary)

After running the simulation, we now want to export all the results which we have created. For this we will export all the reports we can:

sim.ExportReportFile("TutorialReportFile")

sim.ExportInputFile("TutorialInputFile")

sim.ExportRunMessagesFile("TutorialRunMessageFile")

sim.ExportSummaryFile("TutorialSummaryFile")

sim.ExportInputFileWithGraphics("TutorialInputGraphics")

Since we changed the layout of the flowsheet we would also like to save the file under a new name.

sim.SaveAs("AspenfileAfterTutorial.bkp", True)

## Case 1.1: Single variable varying:

The Python file for this tutorial is called CaseExample1.py

Diagram

Description automatically generated

Figure Flowsheet layout

In this case, there is a distillation sequence where the stage numbers are set depending on a list value. Once a value has been set the simulation is run and the output data is found. In this case, the main data points of interest are the reboiler/condenser temperature and duty. These are then plotted against the stage numbers but if the convergence was unsuccessful the value was excepted.

This follows a certain pattern that can be used to loop through any characteristic and find the results. For this, I will present it as a pseudo-code:

*ListOfInputs = []*

*Loop through ListOfInputs:*

*Set Input*

*Run simulation*

*Get outputs*

*Do some data analysis*

The results for the first case are given below and they show the results in dependence of the number of stages.

Chart, line chart

Description automatically generated

Figure split fraction of distillation column in bottom stream per stage number

Chart, line chart

Description automatically generated

Figure Condenser heat duty depending on number of stages

Chart, line chart

Description automatically generated

Figure Reboiler heat duty depending on number of stages

By changing a few lines of code, this kind of analysis could be done for any variable in Aspen Plus. It enables you also to make many simulations to find trends and patterns. This could also be used to learn about the impact of design parameters on your overall design.

## Case 1.2. Double variable varying

This case is almost the same as the previous one just that instead of only going through the list of stage numbers for only the first column you will loop over the columns as well as the stage number list. This is supposed to show how it is possible to make this looping system substantially larger by only adding a few lines of code. If it is planned to do more complex optimizations, it will be necessary to use different methods since the number of possible combinations rises exponentially the more loops are added. To minimize the number of samples needed to find the optimum, we will have to use an optimization algorithm.

Chart, line chart

Description automatically generated

Figure Condenser temperature for two columns depending on stage number

Chart, line chart, histogram

Description automatically generated

Figure Splitfractions in bottoms depending on stage number

As you can see, the split fraction in B4 seems to reach diminishing returns for water by around 12 stages. This means that if our goal is to remove the water this would already suffice. In comparison to that, the B3 still increase its split fraction for water even at 20 stages. This can then be used as an indication that even more stages might be necessary. This is a very simple example for this type of looping through lists and it could be even more interesting by doing this split fraction test for differing pressures and other variables.

## Scipy Optimization library:

The Python file for this tutorial is called CaseExample2.py

Diagram

Description automatically generated

Figure Flowsheet layout

This optimization makes use of the scipy library which includes a large list of options for algorithms to use. In this case, a genetic algorithm and a brute force method will be applied. For this, it is necessary to supply the scipy library with a function that can be called, that intakes certain variables and which returns an integer value. This integer value will be minimized by repeatedly calling this function with varying input values. This minimized value is usually called an objective/loss function since it summarizes the entire design into one parameter.

In the case example this function is named: funcToMin and it is required to do the following things:

1. Unpacks the input variables
2. Sets the input variables into the Aspen Plus flowsheet for both columns
3. Runs the simulation
4. Retrieves outputs and prints them
5. Retrieves values needed for sizing and cost analysis
6. Sizes equipment, Investment cost analysis, revenue, annual profit
7. Returns the loss

The inputs are given in an array called: “v”. In this case it will contain the stage number and the reboiler duty for both columns as the variables which can be altered to arrive at the minimal return. This array can be changed such that it contains any number of inputs which means that it would be easy to add more parameters to the optimization as long as the funcToMin still runs. This means that the only limit for the variables which can be used is Aspen Plus being able to use them in its calculation.

If, for example, one would also want to vary the condenser duty this could be done by following these steps:

1. Add condenser duty to “v” as well as the bounds
2. Add it to the unpacking step in A)
3. Include it in step B)

Besides that, it is always necessary to give certain upper and lower bounds for each variable which is varied in the process of the optimization. These can be used to restrict the number of runs needed to find the optimum as well as restrict Aspen simulation errors. In our case, we chose the arbitrary values of 2-20 and 100-100000 for stages and duties, respectively.

**Loss function:**

The goal of making a column design is usually to find a case where it is possible to make a profit.

Our annual profit is calculated as such:

Annual expected profit = AnnualRevenue – AnnualInvestmentCosts – AnnualUtilityCosts

The revenue is found by taking a certain purity expectation (in this case it is 95%) and checking if any of the product streams have any components at such a purity. If this is the case, then the flow is multiplied by the value of the product.

If purity is enough: AnnualRevenue =

Paying of the loan is an annualized version of the investment costs. Here it is assumed that the bank wants to have the money back within 20 years with this a linear pay schema.

AnnualInvestmentCosts =

By using an expected steam cost of 18$/t and a cooling water price of 0.006$/t it is possible to calculate the costs for the utility by taking the reboiler and condenser duty from the Aspen Plus design.

Since there is a large amount of data for historical costs when developing a new plant, we can make use of this information to achieve an expected investment cost for the design and construction of the columns. This requires an accurate calculation of the size and dimensions which is done by a code supplied by [Christos](https://github.com/ADChristos/Aspen-RL) as well as using correlations to find the expected costs.

**Loss function**

The loss/objective function describes the variable which is to be minimized by the optimization.

Since we wish to maximize the profit the most obvious loss function would be:

Loss =

If a large AnnualExpectedProfit is given this would result in a small loss and if the profit is small this would result in a large loss.

This was initially tried but it failed since it is possible that the design generates a negative profit. In this case the loss would be a small but negative number. If the algorithm evaluates this, it will find that a negative loss (caused by negative profit) would be far preferable to any other number. (since negative<positive) This means that the optimisation algorithm tried to find designs which would be as investment and utility heavy as possible. This is an example of misalignment which can only be repaired by changing the loss function.

New loss function:

Now any profit results in a lower loss compared to negative profit.

### Case 2.1. Genetic algorithm

Genetic algorithms are a type of evolutionary algorithms which are inspired by the biological ideas of mutation, crossover, and selection. This is based on the survival of the fittest, which is commonly used in biology to explain the evolution of a species towards some global optimum with high reliability while still maintaining the possibility of adapting to changed conditions in the future.

For the optimizations in nature, there is a “fitness function” which is synonymous to the long-term maximization of your own genes. In our case, we will use the mathematics that model the biological systems and replace the fitness with negative profit. (because the mathematical model will still maximize it even if it’s no longer about biology but rather “breeding” Aspen flowsheets)

At the beginning of the simulation, there is a parameter called: popsize which describes the number of individuals/organism/instantiations which are initially taken. Each of these has some internal representation (aka chromosomes) to decide on the value of the input variables. The chromosomes which are initially used are usually random. These individuals are then applied to our funcToMin function which will yield a fitness value for everyone. (which would be negative profit) Once these are known for every individual their” generation” is over. Afterward, the reproduction phase starts where the next generation is bred. For this, we will select the most successful individuals and breed them together which mixes their genomes to create a new combination. These two steps are called selection and crossover. If this were to continue for a long time this would result in a homogenous genetic code for each individual where the genes that were initially randomly created would be the only ones that could be selected. To get around this problem, each time a new individual is created there is a chance that some of its genes are randomly set without the influence of its parents.(aka mutation) This randomness will also be counteracting the tendency to land at local minimums.

This process of repeated breeding of individual genomes will at some point result in the group becoming more and more well adapted to the environment. As usually in optimization algorithms the final question is: how long will that take?

Expected total number of runs = (generation number + 1) \* popsize \* len(v)

The selection of popsize, mutation rate, and the other parameters is very difficult since there are no generally applicable heuristics. They are currently just set such that they finish the simulation within half an hour. (each simulation takes around 1 sec)

**Advantages:**

* Widely applicable
* Can be substantially more efficient than brute force (in the right condition)
* Does not guarantee the global minimum

**Disadvantages:**

* Many learning parameters need to be set
* Does not converge cleanly onto one solution

**Results:**

The next problem that you will notice is that the profit of even the most ideal case does not make any money. One needs to consider that there are only two distillation columns that are supposed to split up 5 different compounds which will of course not work. This would be a problem if you want to build this process, but we are just trying to show the process of optimizing a flowsheet.

In the Figure below one can see that with time the algorithm was able to converge onto a solution that is better than the ones at the beginning. There are still some non-ideal choices since the random mutations will result in partially random outputs.

Chart, scatter chart

Description automatically generated

Figure Negative profit per run by genetic algorithm

### Case 2.2. Brute force algorithm:

The brute force method is as it already states a very rough method for finding the optimum. What happens is that the bounds of each variable are subdivided into Ns parts. After this is done for each variable, the algorithm will go through every single combination of all variables. This is an exponential situation since the number of runs will be:

Number of Runs = Ns^len(v)

This problem is called the combinatorial explosion since an already small number of additional variables will result in extremely large search spaces.

It is also possible to activate the “finishing” parameter if nothing is given. This feature will call the brute force method again but on a smaller area of the search space and attempt to find even more detailed optimums around the previously found one. This will be visible in the results section.

**Advantages:**

* Is guaranteed to find the global optimum
* Very simple and applicable in all cases
* Only a few parameters that influence optimization

**Disadvantages:**

* Only rarely an efficient method

**Results:**

As you can see in the figure below the brute force method went through the 4 input variables and varied each of them between the minimum and maximum bounds. Around step 100 he had discovered a local minimum and then started to polish the solution by redefining the min and max bounds around this local minimum and going through all the possibilities. This is repeated until there are insignificant changes in the negative profit at which point it stops. Interestingly this brute force method was much faster at finding the global minimum compared to the genetic code. For this, one needs to consider that the example is very simple, and the exponential curve of possibilities has not kicked in yet.

Chart

Description automatically generated

Figure Negative profit per run by brute force

**Tips and tricks:**

A screenshot of a computer

Description automatically generated with medium confidenceLooping through Aspen node elements (children):  
The code example above allows you to loop through all elements inside of a node even if you do not know the name of the sub nodes. In this case, it loops through all compounds, but it is also possible to do so with all streams, nodes, or ports within a block. This also allows the determination of the names of all streams in the system or all the ports on a block.

Figure Example Code: loop through all Compounds

## Future improvements to the library:

1. Shell to simplify the running of optimizations

This would be equivalent to the step function, but it needs to be generalized such that any possible problem can be included. This would need to be an improved version of Case example 2.

1. Cost analysis needed for optimizing

When optimizing a flowsheet, one needs to have an objective function which can be used to decide which designs are better than others. In our case we would usually take maximized profit and minimized investment costs. To do this one needs to be able to do sizing and costing of equipment, this is sadly not possible over the variable manager which forces us to find other ways.

There are three methods proposed to include a cost analysis into this system, the first would be the custom writing of a sizing and costing function for each piece of equipment. This has been done for the Radfrac equipment by: [(Christos Thesis).](https://github.com/ADChristos/RLforChE) This was used for the optimizations which are seen in the cases.

The second option would be the usage of a Python library that was created by another person. An example of this would be [(ChemEngDPpy)](https://github.com/weepctxb/ChemEngDPpy) who already went through the trouble of creating such a toolbox. The problem with using this library is that one would need to trust the creator of the code for its functionality and trustworthiness and connect the two libraries.

The third option is the easiest and that would be the inclusion of the internal Aspen Plus costing feature. Sadly, there has been no method found to retrieve this information out of the variable manager.

1. Flowsheet-independent Cost analysis

Currently, the cost analysis is done with a known list of product streams and all the Blocks are constant. These conditions do not need to persist this way. With more features like a GetProductStreamNameList() and GetBlocknameList(), it would be possible to write a function that automatically detects what is in the flowsheet and then approximates the costs. These flexibilities are needed to create more layers of abstraction between Aspen and the user.

The same is also the case for higher-level functions such as “calculate profit for the entire flowsheet” and much more.

If you have made such advances or any other advances you can send me an email: [Richardxtenxhagen@gmail.com](mailto:Richardxtenxhagen@gmail.com) and then I can add it to the newest version.

# Power functions

## Flow sheeting, Placing, Removing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Powerfunctions: | | | | |
| **Function description:** | **Function name:** | **Path in Aspen:** | **Function number:** | **Inputs:** |
| Delete Blocks: | BlockDelete() | BLK.Remove(”Blockname”) |  | Blockname |
| Place Blocks: | BlockPlace() | BLK.Add(“Blockname”!TYPE) |  | Blockname and Type = “MATERIAL”, “HEAT” or “” |
| Delete Stream: | StreamDelete() | STRM.Remove(Streamname) |  | Streamname |
| Add stream: | StreamPlace() | STRM.Add(Name !TYPE) |  | Streamname and Type |
| Connect Streams: | StreamConnect() | STRM.Blockname.Ports  .Portname.Add(streamname) |  | Blockname, Streamname, Portname = “D(OUT)”, “B(OUT)” and “F(IN)” |
| Disconnect Streams: | StreamDisconnect() | STRM.Blockname.Ports.  Portname.Remove(streamname) |  | Blockname, Streamname |
| Delete all Streams: | StreamDeleteALL() | STRM.RemoveAll |  |  |
| Delete all Blocks: | BlockDeleteALL() | BLK.RemoveAll |  |  |

## Running Simulations:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description:** | **Function name:** | **Path in Aspen:** | **Function number:** | **Inputs:** |
| Change Visibility: | VisibilityChange() | Aspen.Visibility |  | True or False |
| Return Aspen document name | Give\_AspenDocumentName() | Aspen.Fullname |  |  |
| Dialog Suppression: | DialogSuppression() | Aspen.SuppressDialog |  | True or False |
| Run the Simulation Engine: | EngineRun() | Aspen.Run2() |  |  |
| Stop the Simulation Engine: | EngineStop() | Aspen.Stop() |  |  |
| Reinitiate the Simulation Engine: | EngineReinit() | Aspen.Reinit() |  |  |
| Reinitiate the Block | BlockReinit() |  |  |  |
| Reinitiate the Stream | StreamReinit() |  |  |  |

## Saving Reports, Exporting Files:

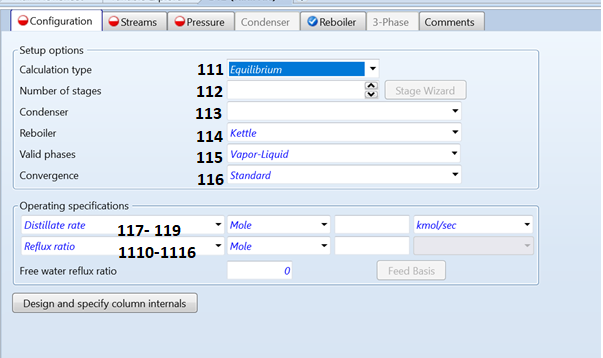
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description:** | **Function name:** | **Path in Aspen:** | **Function number:** | **Inputs:** |
| Save Aspen: | Save() | Aspen.Save() |  |  |
| SaveAs: | SaveAs() | Aspen.SaveAs  (filename,Overwrite) |  | Filename, Overwrite = True |
| Export Backup file: | ExportBackupFile() | Aspen.Export  (1, “filename”) |  | Filename |
| Export Report file: | ExportReportFile() | Aspen.Export  (2, “filename”) |  | Filename |
| Export Summary file: | ExportSummaryFile() | Aspen.Export  (3, “filename”) |  | Filename |
| Export Input file: | ExportInputFile() | Aspen.Export  (4, “filename”) |  | Filename |
| Export RunMessages file: | ExportRun  MessagesFile() | Aspen.Export  (5, “filename”) |  |  |
| Export Flowsheetdrawing file: | ExportFlowsheet  drawingFile() | Aspen.Export  (6, “filename”) |  |  |

# Higher level functions:

|  |  |
| --- | --- |
| **Function description:** | **Function name:** |
| DSTWU | BLK\_DSTWU\_GET\_ME\_ALL\_INPUTS\_BACK |
| BLK\_DSTWU\_SET\_ALL\_INPUTS |
| BLK\_DSTWU\_GET\_OUTPUTS |
| MIXER | BLK\_MIXER\_GET\_ME\_ALL\_INPUTS\_BACK |
| BLK\_MIXER\_SET\_ALL\_INPUTS |
| BLK\_MIXER\_GET\_OUTPUTS |
| HEATER | BLK\_HEATER\_GET\_ME\_ALL\_INPUTS\_BACK |
| BLK\_HEATER\_SET\_ALL\_INPUTS |
| BLK\_HEATER\_GET\_OUTPUTS |
| CISTR | BLK\_CISTR\_GET\_ME\_ALL\_INPUTS\_BACK |
| BLK\_CISTR\_SET\_ALL\_INPUTS |
| BLK\_RCSTR\_GET\_OUTPUTS |
| RPLUG | BLK\_RPLUG\_GET\_ME\_ALL\_INPUTS\_BACK |
| BLK\_RPLUG\_SET\_ALL\_INPUTS |
| BLK\_RPLUG\_GET\_OUTPUTS |
| RADFRAC | BLK\_RADFRAC\_GET\_ME\_ALL\_INPUTS\_BACK |
| BLK\_RADFRAC\_SET\_ALL\_INPUTS |
| BLK\_RADFRAC\_GET\_OUTPUTS |
| FLASH2 | BLK\_FLASH2\_GET\_ME\_ALL\_INPUTS\_BACK |
| BLK\_FLASH2\_SET\_ALL\_INPUTS |
| BLK\_FLASH2\_GET\_OUTPUTS |
| SPLITTER | BLK\_SPLITTER\_GET\_ME\_ALL\_INPUTS\_BACK |
| BLK\_SPLITTER\_SET\_ALL\_INPUTS |
| BLK\_FSPLITTER\_GET\_OUTPUTS |
| RYIELD | BLK\_RYIELD\_GET\_ME\_ALL\_INPUTS\_BACK |
| BLK\_RYIELD\_SET\_ALL\_INPUTS |
| BLK\_RYIELD\_GET\_OUTPUTS |
| STRM | STRM\_GET\_ME\_ALL\_INPUTS\_BACK |
| STRM\_SET\_ALL\_INPUTS |
| STRM\_GET\_OUTPUTS |

# RADFRAC Inputs

## Configuration: Page 1



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Calculation Type | BLK\_RADFRAC\_Set  \_CalculationType | Input.CALC\_MODE  .Value | 111 | 1. RIG-RATE 2. EQUILIBRIUM |
| Number of Stages in Column | BLK\_RADFRAC\_Set\_  NSTAGE | Input.NSTAGE.Value | 112 |  |
| Condenser type: | BLK\_RADFRAC\_Set  \_CondenserType | Input.CONDENSER  .Value | 113 | 1. ”NONE” 2. ”TOTAL” 3. ”PARTIAL-V” 4. ”PARTIAL-V-L” |
| **DETERMINES THE CONDENSER INPUT VARIABELS NEEDED ON PAGE 4** | | | | |
| Reboiler type: | BLK\_RADFRAC\_Set  \_CondenserType | Input.REBOILER.Value | 114 | 1. ”NONE” 2. ”KETTLE” 3. ”THERMOSYPHON” |
| **CHANGES THE CONDENSER INPUT VARIABELS NEEDED ON PAGE 5** | | | | |
| ValidPhase: | BLK\_RADFRAC\_Set\_  Phases | Input.PHASE.Value  Input.NPHASE.Value | 115 | 1. “V” 2. “L” 3. “S”   And  1,2 or 3 |
| Convergence method | BLK\_RADFRAC\_Set\_  ConvergenceMethod | Input.CONV\_METH  .Value | 116 | 1. ”STANDARD” 2. ”PETROLEUM” 3. ”NONIDEAL” 4. ”AZEOTROPIC” 5. ”CRYOGENIC” 6. ”OTHERS” |
| Refluxratio | BLK\_RADFRAC\_Set\_  Refluxratio | Input.BASIS\_RR.Value | 117 |  |
| Refluxrate | BLK\_RADFRAC\_Set\_  Refluxrate | Input.BASIS\_L1.Value | 118 |  |
| Boilup flowrate | BLK\_RADFRAC\_Set\_  BoilupRate | Input.BASIS\_VN.Value | 119 |  |
| Boilup ratio (Boilup rate/Bottoms rate) | BLK\_RADFRAC\_Set\_  BoilupRatio | Input.BASIS\_BR.Value | 1110 |  |
| Condenser duty (negative means Q removal) | BLK\_RADFRAC\_Set\_  CondenserDuty | Input.Q1.Value | 1111 |  |
| Reboiler duty | BLK\_RADFRAC\_Set\_  ReboilerDuty | Input.QN.Value | 1112 |  |
| Total distillate flowrate | BLK\_RADFRAC\_Set\_  TotalDestillateFlowrate | Input.BASIS\_D.Value | 1113 |  |
| Liquid bottoms flow rate | BLK\_RADFRAC\_Set\_  LiquidBottomRate | Input.BASIS.B.Value | 1114 |  |
| Destillate to Feed flow ratio | BLK\_RADFRAC\_Set\_  DestillateToFeedRatio | Input.BASIS.D:F.Value | 1115 |  |
| Bottom to Feed flow ratio | BLK\_RADFRAC\_Set\_  BottomToFeedRatio | Input.BASIS.B:F.Value | 1116 |  |

## Streams Page 2

Graphical user interface, application, table

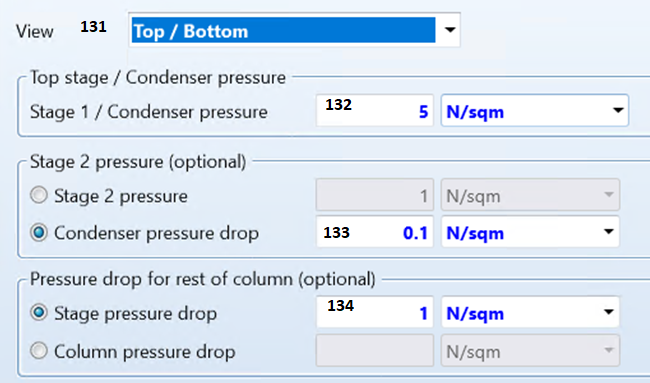
Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Feed stage for Feed | LK\_RADFRAC\_Set\_  FeedStage | Input.FEED\_STAGE  .Feedname.Value | 121 | Feedname |
| Feed stage Location | BLK\_RADFRAC\_Set  \_FeedStageLocation | Input.FEED\_CONVE2.  Feedname.Value | 122 | 1. ON-STAGE 2. ABOVE-STAGE 3. ON-STAGE-VAP 4. ON-STAGE-LIQ   Feedname |
| Product stage number | BLK\_RADFRAC\_Set  \_ProductStreamStage | Input.PROD\_STAGE.  Feedname.Value | 123 | Feedname |
| Product stream Phase: | BLK\_RADFRAC\_Set  \_ProductPhase | Input.PROD\_PHASE.  Feedname.Value | 124 | 1. L (Liquid) 2. L1 (1st liquid) 3. L2 (2nd liquid) 4. W (??) 5. V (Vapor) 6. TL (Total liquid) 7. TV (Total vapor)   Feedname |

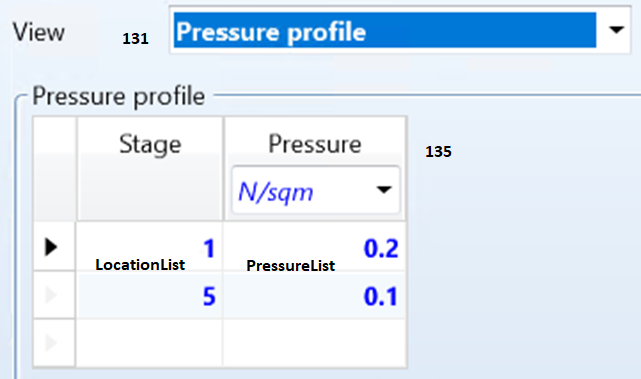
## Pressures Page 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Pressure input perspective | BLK\_RADFRAC\_Set  \_Pressure  PerspectiveOption | Input.VIEW\_PRES  .Value | 131 | 1. TOP/BOTTOM 2. PROFILE 3. PDROP |
| **If you chose TOP/BOTTOM:** | | | | |
| Condenser pressure | BLK\_RADFRAC\_Set\_  TOPBOTTOM\_  CondenserPressure | Input.PRES1.Value | 132 |  |
| Condenser pressure drop | BLK\_RADFRAC\_Set\_  TOPBOTTOM\_  CondenserPressure  Drop | Input.PRES2.Value | 133 |  |
| Pressure drop per stage | BLK\_RADFRAC\_Set\_  TOPBOTTOM  \_StagePressureDrop | Input.DP\_STAGE  .Value | 134 |  |
| **If you chose PROFILE** | | | | |
| Set pressure profile | BLK\_RADFRAC\_Set\_  PROFILE\_Pressure | Input.STAGE\_PRES.**5**.Value = 0.1 | 135 | PressureList=[0.2,0.1] LocationList=[1,5] |
| **If you chose PDROP** | | | | |
| Top stage Pressure: | BLK\_RADFRAC\_Set\_  PDROP\_TopStage  Pressure | Input.PRES1.Value | 136 |  |
| Defining the PDrop profile | BLK\_RADFRAC\_Set\_  PDROP\_StagePDROP\_Profile | Input.PRES\_  STAGE1.**1**.Value = 2 | 137 | LocationList =[1,2]  StartingStageList=[2,7]  EndingStageList =[6,10]  PressureDropList=[0.1,0.2] |

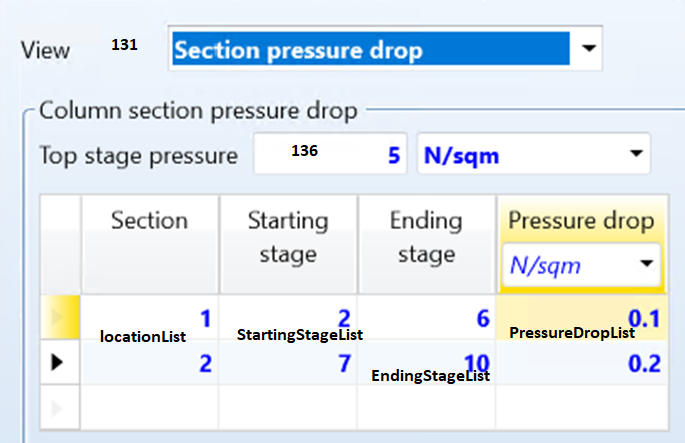
IF TOP/BOTTOM is selected: only Condenser pressure is required for calculation



If Pressure PROFILE is selected:



If PDROP is selected:

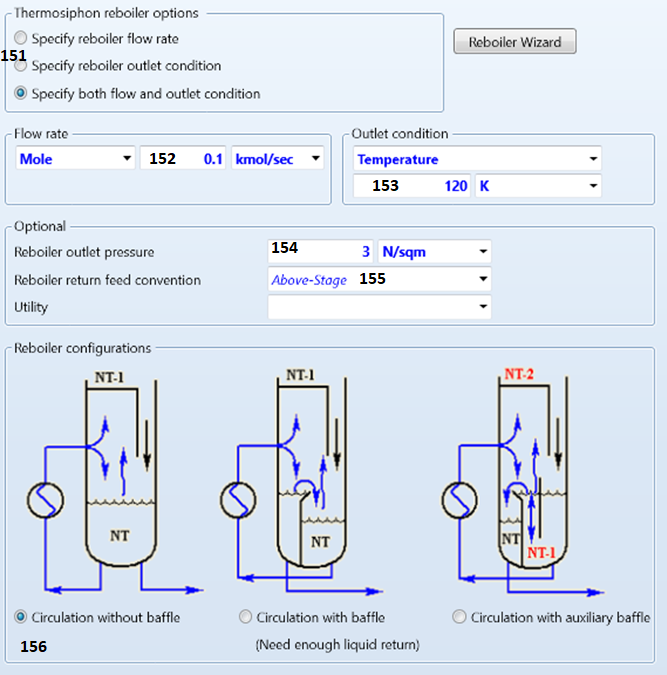


## Condenser Page 4



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| **If NONE was chosen for Condenser Type: NO INPUTS NEEDED** | | | | |
| **If TOTAL or PARTIAL-V was chosen:** | | | | |
| Option for Subcooling specs is needed | BLK\_RADFRAC\_Set\_  TOTALorPARTIALV\_CondenserTempOption | Input.OPT\_SUBCOOL  .Value | 141 | 1. TEMP 2. SUBCOOL |
|  | **If TEMP was chosen:** | | | |
| Temperature | BLK\_RADFRAC\_Set\_  TOTALorPARTIALV\_TEMP | Input.SC\_TEMP.Value | 142 |  |
|  | **If SUBCOOL was chosen:** | | | |
| Degrees Subcooled | BLK\_RADFRAC\_Set\_  TOTALorPARTIALV\_SUBCOOL | Input.DEGSUB.Value | 143 |  |
| Destillate subcooling Option | BLK\_RADFRAC\_Set\_  TOTALorPARTIALV\_  CoolRefluxandDestillate | Input.SC\_OPTION  .Value | 144 | 1. REFLUX-AND-DESTILLATE 2. REFLUX-ONLY |
| **if PARTIAL\_V\_L was chosen:** | | | | |
| Condenser Temperature Option | BLK\_RADFRAC\_Set\_PARTIAL\_V\_L\_CondenserTempOption | Input.OPT\_SUBCOOL  .Value | 141 | 1. TEMP 2. SUBCOOL |
|  | **If TEMP was chosen for TempOption:** | | | |
| Temperature | BLK\_RADFRAC\_Set  \_PARTIAL\_V\_L\_TEMP | Input.SC\_TEMP.Value | 142 |  |
|  | **If SUBCOOL was chosen for TempOption:** | | | |
| Degrees subcooled | BLK\_RADFRAC\_Set\_PARTIAL\_V\_L\_SUBCOOL | Input.DEGSUB.Value | 143 |  |
| Destillate subcooling Option | BLK\_RADFRAC\_Set\_PARTIAL\_V\_L\_CoolRefluxandDestillate | Input.SC\_OPTION  .Value | 144 | 1. REFLUX-AND-DESTILLATE 2. REFLUX-ONLY |
| Condenser Option | BLK\_RADFRAC\_Set\_PARTIAL\_V\_L\_CondenserOption | Input.OPT\_COND  .Value | 145 | 1. TEMP 2. VFRAC |
|  | **If you chose TEMP for Condenser Option:** | | | |
| Vapor Temp | BLK\_RADFRAC\_Set\_PARTIAL\_V\_L\_TEMP\_VaporTemp | Input.T1.Value | 146 |  |
|  | **If you chose VFRAC for Condenser Option:** | | | |
| Vapor fraction | BLK\_RADFRAC\_Set\_PARTIAL\_V\_L\_TEMP\_VaporFraction | Input. BASIS\_RDV.Value | 147 |  |

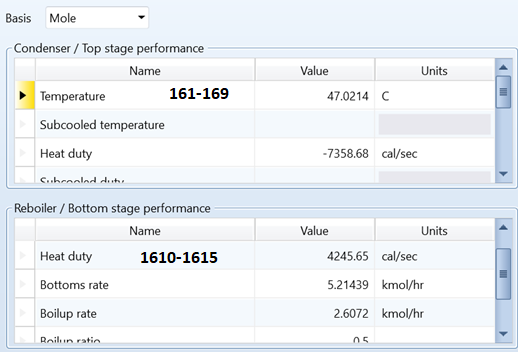
## Reboiler Page 5



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | | **Extra Inputs** |
| **If you chose NONE or KETTLE on Page 1: no inputs needed** | | | | | |
| **If you chose THERMOSYPHON** | | | | | |
| Thermosyphon Options: | BLK\_RADFRAC\_Set\_  THERMOSYPHON\_OPTIONS | Input.OPT\_TH\_REB.Value | 151 | 1. FLOW 2. OUTLET 3. FLOW+OUTLET | |
|  | **If FLOW was selected:** | | | | |
| Set Flow | BLK\_RADFRAC\_Set\_  THERMOSYPHON\_FLOW | Input.TH\_FLOW  .Value | 152 |  | |
|  | **If OUTLET was selected:** | | | | |
| Set Outlet temp | BLK\_RADFRAC\_Set\_  THERMOSYPHON\_OUTLET | Input.TH\_TEMP  .Value | 153 |  | |
|  | **IF FLOW+OUTLET was selected:** | | | | |
| Set Flow | BLK\_RADFRAC\_Set\_  THERMOSYPHON\_FLOW | Input.TH\_FLOW  .Value | 152 |  | |
| Set Outlet temp | BLK\_RADFRAC\_Set\_  THERMOSYPHON\_OUTLET | Input.TH\_TEMP  .Value | 153 |  | |
| **Optional Parameters:** | | | | | |
| Outlet pressure | BLK\_RADFRAC\_Set\_  ReboilerOutletPressure | Input.TH\_PRES  .Value | 154 |  | |
| Reboiler return location | BLK\_RADFRAC\_Set\_  ReboilerReturnLocation | Input.RETURN\_  CONV.Value | 155 | 1. ABOVE-STAGE 2. ON-STAGE | |
| Reboiler configuration | BLK\_RADFRAC\_Set\_  ReboilerConfiguration | Input.TSR  \_CONFIG.Value | 156 | 1. 1 2. 2 3. 3 | |

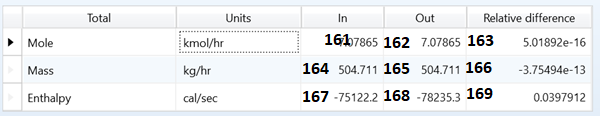
# RADFRAC OUTPUTS

## Summary: Page 1 Output



|  |  |  |  |
| --- | --- | --- | --- |
| **RADFRAC Outputs Page 1: Summary** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| **Condenser Data:** | | | |
| Condenser Temperature | BLK\_RADFRAC\_Get\_Condenser\_Temperature | Output. TOP\_TEMP | 161 |
| Subcooled temperature | BLK\_RADFRAC\_Get\_Condenser\_SubcooledTemp | Output.SCTEMP | 162 |
| Heat duty: | BLK\_RADFRAC\_Get\_Condenser\_HeatingDuty | Output.COND\_DUTY | 163 |
| Subcooled duty: | BLK\_RADFRAC\_Get\_Condenser\_SubcooledDuty | Output.SCDUTY | 164 |
| Distillate rate: | BLK\_RADFRAC\_Get\_Condenser\_DistillateRate | Output.MOLE\_D | 165 |
| Reflux rate: | BLK\_RADFRAC\_Get\_Condenser\_RefluxRate | Output.MOLE\_L1 | 166 |
| Free water distillate rate: | BLK\_RADFRAC\_Get\_Condenser\_  FreeWaterDistillateRate | Output.MOLE\_DW | 167 |
| Free water reflux ratio: | BLK\_RADFRAC\_Get\_Condenser\_  FreeWaterRefluxRatio | Output.RW | 168 |
| Distillate to feed ratio: | BLK\_RADFRAC\_Get\_Condenser  \_DistillateToFeedRatio | Output.MOLE\_DFR | 169 |
| **Reboiler Data:** | | | |
| Temperature: | BLK\_RADFRAC\_Get\_Reboiler\_Temperature | Output.BOTTOM\_TEMP | 1610 |
| Heat duty: | BLK\_RADFRAC\_Get\_Reboiler\_HeatDuty | Output.REB\_DUTY | 1611 |
| Bottoms rate: | BLK\_RADFRAC\_Get\_Reboiler\_BottomsRate | Output.MOLE\_B | 1612 |
| Boilup rate: | BLK\_RADFRAC\_Get\_Reboiler\_BoilupRate | Output.MOLE\_VN | 1613 |
| Boilup ratio: | BLK\_RADFRAC\_Get\_Reboiler\_BoilupRatio | Output.CMF\_MAMX | 1614 |
| Bottoms to feed ratio: | LK\_RADFRAC\_Get\_Reboiler  \_BottomsToFeedRatio | Output.MOLE\_BFR | 1615 |

## Balance: Page 2 Output

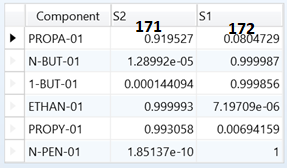


|  |  |  |  |
| --- | --- | --- | --- |
| **RADFRAC Outputs Page 2: Balance** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Mole Balance: IN | BLK\_RADFRAC\_Get\_MoleFlowBalanceIN() | Output.BAL\_MOLI\_TFL | 161 |
| Mole Balance: OUT | BLK\_RADFRAC\_Get\_  MoleFlowBalanceOUT() | Output.BAL\_MOLO\_TFL | 162 |
| Mole Balance: Relative difference | BLK\_ RADFRAC\_Get\_  MoleFlowBalanceRelDifference() | Output.BAL\_MOLR\_TFL | 163 |
| Mass Balance: IN | BLK\_ RADFRAC\_Get\_  MassFlowBalanceIN() | Output.BAL\_MASI\_TFL. | 164 |
| Mass Balance: OUT | BLK\_ RADFRAC\_Get\_  MassFlowBalanceOUT() | Output.BAL\_MASO\_TFL | 165 |
| Mass Balance: Relative difference | BLK\_ RADFRAC\_Get\_  MassFlowBalanceRelDifference() | Output.BAL\_MASR\_TFL | 166 |
| Enthalpy: IN | BLK\_ RADFRAC\_Get\_EnthalpyBalanceIN() | Output.TOT\_ENTH\_ABS | 167 |
| Enthalpy: OUT | BLK\_RADFRAC\_Get\_EnthalpyBalanceOUT() | Output.BAL\_ENTH\_OUT | 168 |
| Enthalpy: Relative difference | BLK\_ RADFRAC\_Get\_  EnthalpyBalanceRelDifference() | Output.TOT\_ENTH\_REL | 169 |

## Split fractions: Page 3 Output

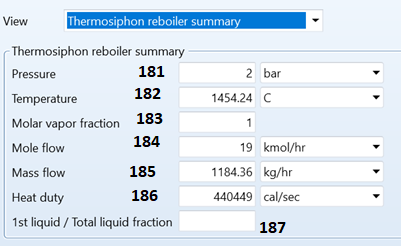
**Table

Description automatically generated**

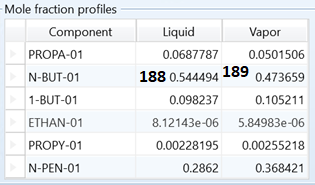


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RADFRAC Outputs Page 3: Split fractions** | | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** | **Inputs** |
| Splitfraction in Stream1 | BLK\_RADFRAC\_Get\_SplitFractionInS1 | Output.MASS\_CONC  .Compoundname.S1 | 171 | Compound name |
| Splitfraction in Stream2 | BLK\_RADFRAC\_Get\_SplitFractionInS2 | Output.MASS\_CONC  .Compoundname.S2 | 172 | Compound name |

## Reboiler: Page 4 Output



|  |  |  |  |
| --- | --- | --- | --- |
| **RADFRAC Outputs Page 4: Reboiler** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Pressure: | BLK\_RADFRAC\_Get\_  Thermosiphon\_Pressure | Output.TH\_PRES\_OUT | 181 |
| Temperature: | BLK\_RADFRAC\_Get\_  Thermosiphon\_Temperature | Output.TH\_TEMP\_OUT | 182 |
| Molar vapor fraction: | BLK\_RADFRAC\_Get\_  Thermosiphon\_MolarVaporFraction | Output.TH\_VFRAC\_OUT | 183 |
| Molar flow: | BLK\_RADFRAC\_Get\_  Thermosiphon\_MolarFlow | Output.TH\_MOLEFLOW | 184 |
| Mass flow: | BLK\_RADFRAC\_Get\_  Thermosiphon\_MassFlow | Output.TH\_MASSFLOW | 185 |
| Heat duty: | BLK\_RADFRAC\_Get\_  Thermosiphon\_HeatDuty | Output.TH\_DUTY | 186 |
| First liquid by Total liquid ratio: | BLK\_RADFRAC\_Get\_  Thermosiphon\_FirstliquidByTotalLiquidRatio | Output. LIQ\_RATIO | 187 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RADFRAC Outputs Page 4: Mole fraction profiles** | | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** | **Input** |
| Liquid Reboiler fraction | BLK\_RADFRAC\_Get\_  ReboilerMoleFractionInLiquid | Output.TH\_X  .Compoundname.Value | 188 | Compound name |
| Vapor Reboiler fraction | BLK\_RADFRAC\_Get  \_ReboilerMoleFractionInVapor | Output.TH\_Y  .Compoundname.Value | 189 | Compound name |

## Utilities: Page 5 Output

Graphical user interface, application

Description automatically generated

Not implemented

## Stage Utilities: Page 6 Output

Table

Description automatically generated

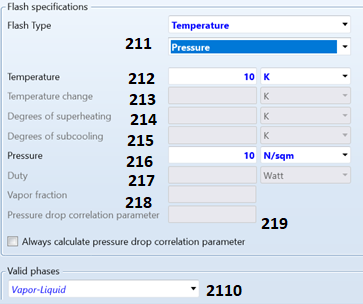
Not implemented

## Status: Page 7 Output

|  |  |  |  |
| --- | --- | --- | --- |
| **RADFRAC Outputs Page 7: Status** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Convergence status: | BLK\_RADFRAC\_Get\_ConvergenceStatus | Output.BLKSTAT.Value = 0 | 191 |
| Convergence Message: | BLK\_RADFRAC\_Get\_ConvergenceMessage | Output.BLKMSG.Value | 192 |
| Property status: | BLK\_RADFRAC\_Get\_PropertyStatus | Output.PROPSTAT.Value=0 | 193 |

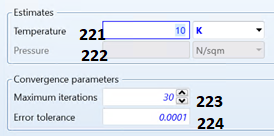
# Heater Input

## Specifications: Input Page 1



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Flash Type | BLK\_HEATER\_Set  \_FlashTypeOption | Input.SPEC\_OPT.Value | 211 | **See Below:** |
| Temperature | BLK\_HEATER\_Set  \_Temperature | Input.TEMP.Value | 212 |  |
| Temp change | BLK\_HEATER\_Set  \_TemperatureChange | Input.DELT.Value | 213 |  |
| Degree superheated | BLK\_HEATER\_Set  \_DegreesSuperheating | Input.DEGSUP.Value | 214 |  |
| Degrees subcooled | BLK\_HEATER\_Set  \_DegreesSubcooling | Input.DEGSUB.Value | 215 |  |
| Pressure | BLK\_HEATER\_Set  \_Pressure | Input.PRES.Value | 216 |  |
| Duty | BLK\_HEATER\_Set  \_Duty | Input.DUTY.Value | 217 |  |
| Vapor fraction | BLK\_HEATER\_Set  \_Vaporfraction | Input.VFRAC.Value | 218 |  |
| Pressure drop correlation parameter | BLK\_HEATER\_Set  \_PressureDropCorrelation | Input.DPPARM.Value | 219 |  |
| Valid Phases | BLK\_HEATER\_Set  \_Phases | Input.PHASE.Value  Input.NPHASE.Value | 2110 | 1. “V” 2. “L” 3. “S”   And  1,2 or 3 |

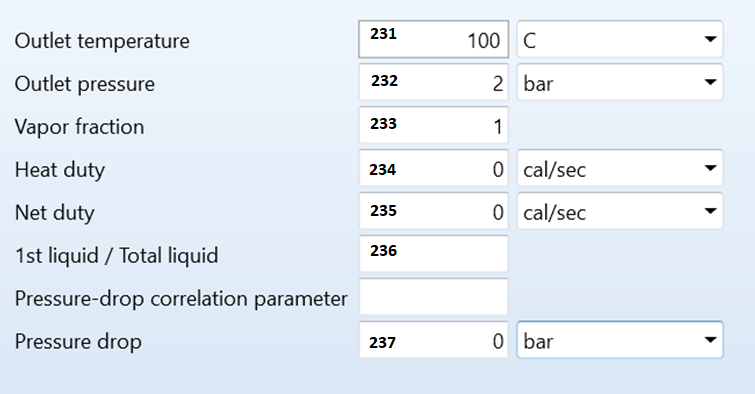
## Flash Options Input Page 2:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Estimated Temperature | BLK\_HEATER\_Set\_  TemperatureEstimation | Input.T\_EST.Value | 221 |  |
| Estimated Pressure | BLK\_HEATER\_Set\_  PressureEstimation | Input.P\_EST.Value | 222 |  |
| Maximum iterations | BLK\_HEATER\_Set\_  MaximumIteration | Input.MAXIT.Value | 223 |  |
| Error tolerance | BLK\_HEATER\_Set\_  ErrorTolerance | Input.TOL.Value | 224 |  |

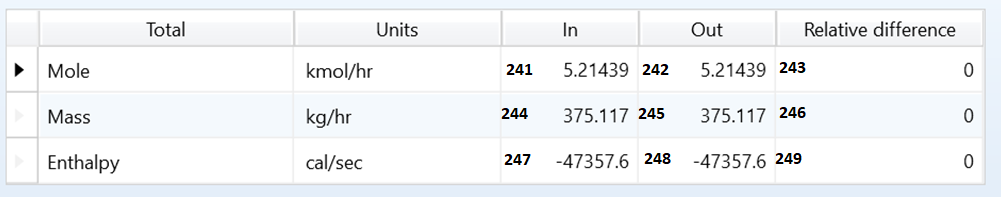
# Heater Output

## Summary Page 1



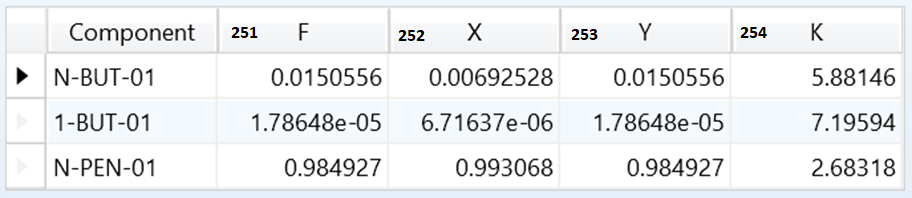
|  |  |  |  |
| --- | --- | --- | --- |
| **Mixer Outputs Page 1: Summary** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Outlet temperature | BLK\_HEATER\_Get\_OutletTemperature | Output.B\_TEMP | 231 |
| Outlet pressure | BLK\_HEATER\_Get\_OutletPressure | Outlet.B\_PRES | 232 |
| Vapor fraction | BLK\_HEATER\_Get\_VaporFraction | Outlet.B\_VFRAC | 233 |
| First liquid by total liquid | BLK\_HEATER\_Get\_FirstLiquidbyTotalLiquid | Outlet.LIQ\_RATIO | 236 |
| Pressure drop | BLK\_HEATER\_Get\_PressureDrop | Outlet.PDROP | 237 |

## Balance Page 2



|  |  |  |  |
| --- | --- | --- | --- |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Mole Balance: IN | BLK\_ HEATER \_Get  \_MoleFlowBalanceIN() | Output.BAL\_MOLI\_TFL.Value | 241 |
| Mole Balance: OUT | BLK\_ HEATER \_Get  \_MoleFlowBalanceOUT() | Output.BAL\_MOLO\_TFL.Value | 242 |
| Mole Balance: Relative difference | BLK\_ HEATER \_Get  \_MoleFlowBalanceRelDifference() | Output.BAL\_MOLR\_TFL.Value | 243 |
| Mass Balance: IN | BLK\_ HEATER \_Get  \_MassFlowBalanceIN | Output.BAL\_MASI\_TFL.Value | 244 |
| Mass Balance: OUT | BLK\_ HEATER \_Get  \_MassFlowBalanceOUT | Output.BAL\_MASO\_TFL.Value | 245 |
| Mass Balance: Relative difference | BLK\_ HEATER \_Get  \_MassFlowBalanceRelDifference | Output.BAL\_MASR\_TFL.Value | 246 |
| Enthalpy: IN | BLK\_ HEATER \_Get  \_EnthalpyBalanceIN | Output.TOT\_ENTH\_ABS.Value | 247 |
| Enthalpy: OUT | BLK\_ HEATER \_Get  \_EnthalpyBalanceOUT | Output.BAL\_ENTH\_OUT.Value | 248 |
| Enthalpy: Relative difference | BLK\_ HEATER \_Get  \_EnthalpyBalanceRelDifference | Output.TOT\_ENTH\_REL.Value | 249 |

## Phase equilibrium Page 3



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** | **Inputs** |
| Total Flow fraction | BLK\_ HEATER\_Get\_  TotalFlowFraction() | Output.F.”Compoundname”.Value | 251 | Compound name |
| Liquid concentration: | BLK\_ HEATER\_Get\_  Liquidconcentrations | Output.X.”Compoundname”.Value | 252 | Compound name |
| Vapor concentration: | BLK\_ HEATER\_Get\_  Vaporconcentrations | Output.Y.”Compoundname”.Value | 253 | Compound name |
| Equilibrium constant: | BLK\_HEATER\_Get\_  EquilibriumConstant | Output.B\_K.”Compoundname”.Value | 254 | Compound name |

## Utilities Page 4

Graphical user interface, application

Description automatically generated

Not implemented yet

## Status Page 5

Graphical user interface, text, application

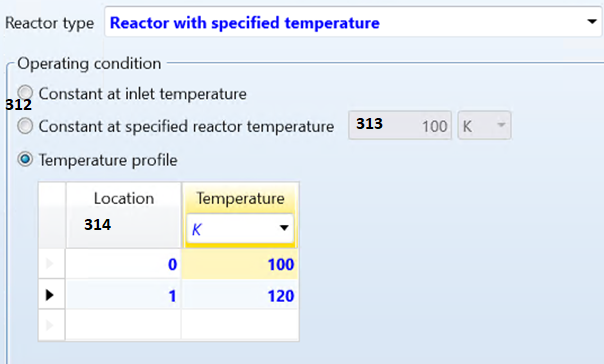
Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Convergence status: | BLK\_HEATER\_Get\_ConvergenceStatus | Output.BLKSTAT.Value = 0 | 261 |
| Convergence Message: | BLK\_HEATER\_Get\_ConvergenceMessage | Output.BLKMSG.Value | 262 |
| Property status: | BLK\_HEATER\_Get\_PropertyStatus | Output.PROPSTAT.Value = 0 | 263 |

# RPLUG Input

## Reactor type Input Page 1:

A) If T-SPEC is selected: Reactor maintains Temperature



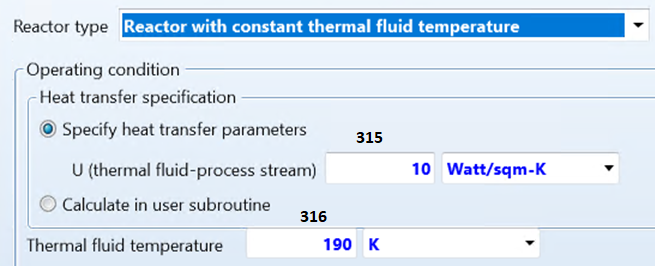
B) if ADIABATIC is selected: Reactor is thermally insulated

Graphical user interface, text, application, email

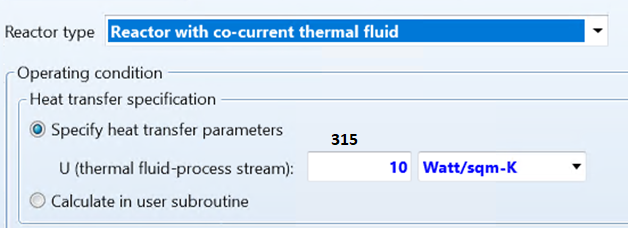
Description automatically generated

Nothing needs to be specified here.

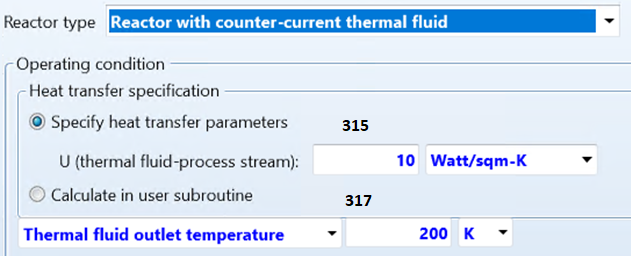
C) if TCOOL-SPEC is selected: specified constant thermal fluid



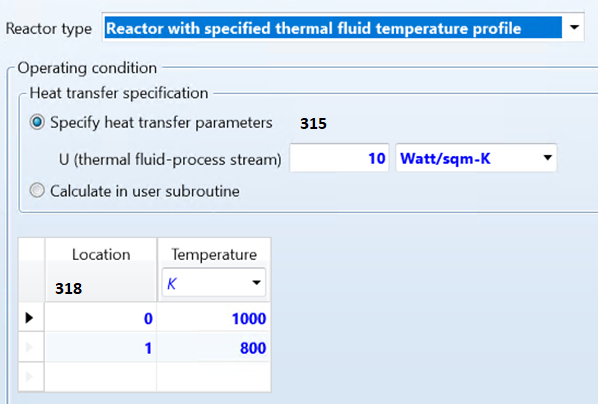
D) if CO-COOL is selected: Co current flow Process Stream



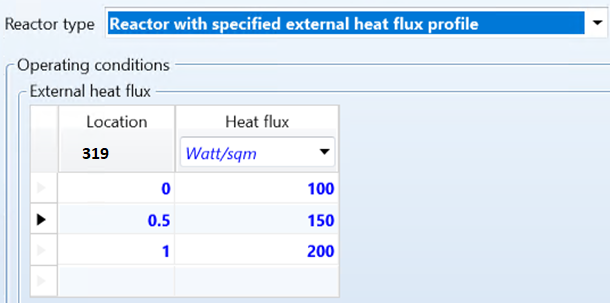
E) if COUNTER-COOL is selected: Counter current flow of thermal fluid



F) if TCOOL-PROF is selected: with defined Temperature Profile

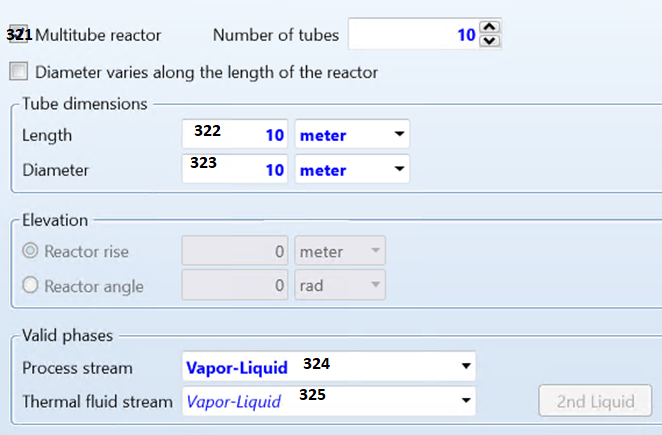


G) if QFLUX-PROF is selected: with defined heat flux profile



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Reactor type | BLK\_RPLUG\_Set\_TYPE | Input.TYPE.Value | 311 | 1. “T-SPEC” 2. ”ADIABATIC” 3. ”TCOOL-SPEC” 4. “CO-COOL” 5. “TCOOL-PROF” 6. “QFLUX-PROF” |
| **If T-SPEC was chosen:** | | | | |
| Operating condition Option: | BLK\_RPLUG\_Set\_T\_SPEC  \_Operating\_condition | Input.OPT\_TSPEC.Value | 312 | 1. INLET-TEMP 2. CONST-TEMP 3. TEMP-PROF |
|  | **If INLET-TEMP was chosen: no input needed** | | | |
|  | **If CONST-TEMP was chosen:** | | | |
| Constant Temperature | BLK\_RPLUG\_Set\_T\_SPEC  \_Constant\_Temp | Input.REAC\_TEMP.Value | 313 |  |
|  | **If TEMP-PROF was chosen:** | | | |
| Setting Temperature Profile | BLK\_RPLUG\_Set\_T\_SPEC\_  TemperatureProfil | Input.LOC.1.Value = 1 | 314 | TemperatureList  LocationList |
| **If ADIABATIC was chosen: no input needed** | | | | |
| **If TCOOL-SPEC was chosen:** | | | | |
| Heat transfer U | BLK\_RPLUG\_Set\_TCOOL\_SPEC\_  HeattransferU | Input.U.Value | 315 |  |
| Constant Temperature | BLK\_RPLUG\_Set\_TCOOL\_  SPEC\_ConstantTemp | Input.CTEMP.Value | 316 |  |
| **If CO-COOL was chosen:** | | | | |
| Heat transfer U | BLK\_RPLUG\_Set\_CO\_COOL\_  HeattransferU | Input.U.Value | 315 |  |
| **If COUNTER-COOL was chosen:** | | | | |
| Heat transfer U | BLK\_RPLUG\_COUNTER\_COOL  \_HeattransferU | Input.U.Value | 315 |  |
| Outlet Temperature | BLK\_RPLUG\_Set\_COUNTER  \_COOL\_OutletTemp | Input.TEMP | 317 |  |
| **If TCOOL-PROF was chosen:** | | | | |
| Heat transfer U | BLK\_RPLUG\_Set\_TCOOL  \_PROF\_HeattransferU | Input.U.Value | 315 |  |
| Temperature Profile | BLK\_RPLUG\_Set\_TCOOL\_  PROF\_TemperatureProfil | Input.TCOOL.#0.Value = 1000 | 318 | TemperatureList= [1000,800]  LocationList= [0,1] |
| **If QFLUX-PROF was chosen:** | | | | |
| Heat flux profile | BLK\_RPLUG\_Set\_QFLUX\_  PROF\_HeatFluxProfil | Input.QFLUX.#0.Value = 100 | 319 | HeatFluxList=  [100, 150, 200]  LocationList = [0,0.5,1] |

## General Reactor Configurations Input Page 2:

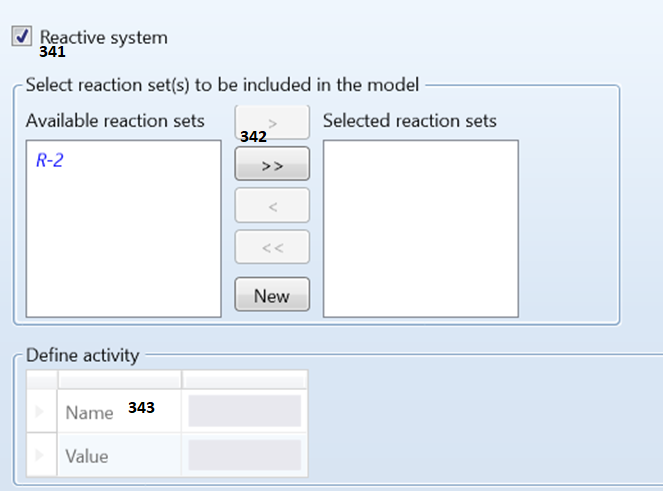


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Activate Multitubular design | BLK\_RPLUG\_Set\_Activate\_  Multitube\_Reactor |  | 321 | Activate\_YES\_NO  Number\_of\_Tubes |
| Tube length | BLK\_RPLUG\_Set\_TubeLength |  | 322 |  |
| Tube Diameter | BLK\_RPLUG\_Set\_TubeDiameter |  | 323 |  |
| Stream Phase | BLK\_RPLUG\_Set\_Phases |  | 324 | Phase number   1. 1 2. 2 3. 3   Phase   1. V 2. L |
| Phase of Thermal fluid | BLK\_RPLUG\_Set\_  Thermalfluid\_ValidPhases |  | 325 | Phase number   1. 1 2. 2 3. 3   Phase   1. V 2. L |

## Streams: Input Page 3:

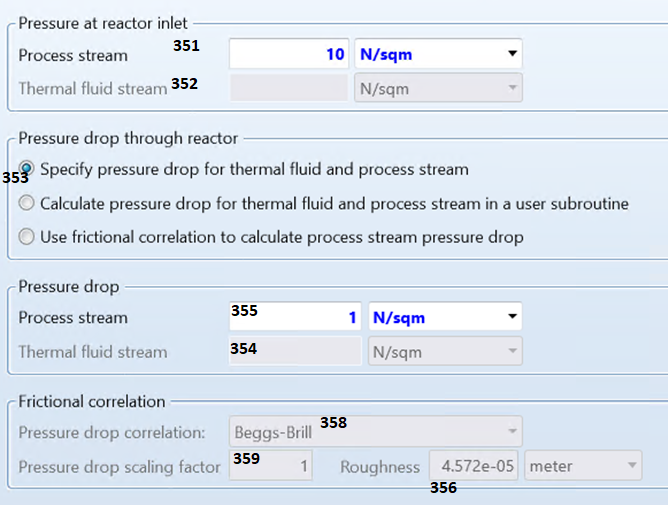
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Productstream  phase | BLK\_RPLUG\_Set\_  Productstream\_phase | Input.PROD\_PHASE.  ”STREAMNAME”.Value | 331 | Streamname  Possible Values:   1. V 2. L 3. L1 4. L2 5. W 6. VL 7. VL1 8. LW 9. L1L2 |

## Reactions: Input Page 4:



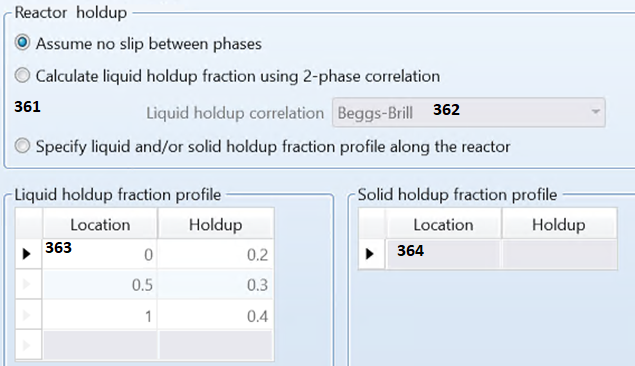
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Activate Reactive System | BLK\_RPLUG\_Set\_  ActivateReactions | Input.REACSYS.Value | 341 | 1. YES 2. NO |
| Add activities: | BLK\_RPLUG\_Set  \_ReactionActivities | Input.ACT\_VALUE.#0.Value | 343 | 1. ActivityList 2. ActivityNameList |

## Pressure specification Input Page 5:



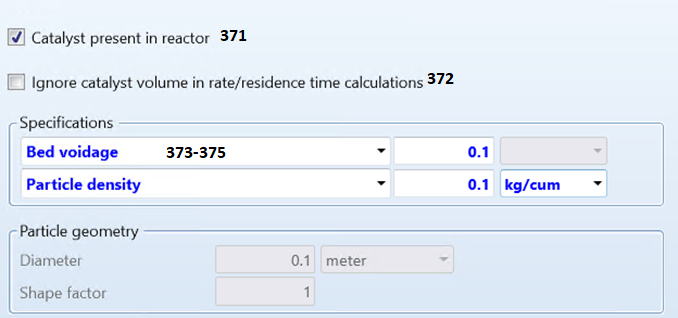
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Reactor Inlet Processflow Pressure | BLK\_RPLUG\_Set\_  InletProcessflow  Pressure | Input.PRES | 351 |  |
| Pressure at inlet for Thermal Fluid stream | BLK\_RPLUG\_Set  \_InletThermalfluid  Pressure | Input.CPRES | 352 |  |
| Choose Pressure drop calculation Option | BLK\_RPLUG\_Set\_  Pressuredrop  CalulationOption | Input.OPT\_PDROP  .Value | 353 | 1. “SPECIFIED“ 2. “USER-SUBR“ 3. “CORRELATION“ |
| **If SPECIFIED is chosen:** | | | | |
| Thermal fluid pressure drop | BLK\_RPLUG\_Set\_  SPECIFIED\_Thermalfluid  PressureDrop | Input.CPDROP  .Value | 354 |  |
| Process flow pressure drop | BLK\_RPLUG\_Set\_  SPECIGIED\_Processflow  PressureDrop | Input.PDROP.Value | 355 |  |
| **If USER-SUBR is chosen:** | | | | |
| Roughness value for USERSUBR | BLK\_RPLUG\_Set\_  USERSUBR\_Roughnessvalue | Input.ROUGHNESS  .Value = 4.572E-05 | 356 |  |
| **If Correlation is chosen:** | | | | |
| Pressure drop of Thermalfluid | BLK\_RPLUG\_Set\_  CORRELATION\_  Thermalfluid  PressureDrop | Input.CPDROP | 357 |  |
| Pressure drop correlation | BLK\_RPLUG\_Set\_  CORRELATION\_  Pressuredrop  Correlation | Input.DP\_FCOR  .Value | 358 | 1. BEGGS-BRILL 2. DUKLER 3. SLACK 4. ORKI 5. AWR 6. LOCK-MART 7. H-BROWN 8. DARCY 9. ERGUN 10. HTFS |
| Correction factor | BLK\_RPLUG\_Set\_  CORRELATION  \_CorrectionFactor | Input.DP\_MULT  .Value = 1 | 359 |  |

## Reactor holdup Input Page 6:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Option to choose the Hold up |  | Input.OPT\_HOLDUP.Value | 361 | 1. NO-SLIP 2. CALCULATED 3. SPECIFIED: |
| **if you chose NO-SLIP: no imputs needed** | | | | |
| **if you chose CALCULATED:** | | | | |
| Liquid holdup correlation | BLK\_RPLUG\_Set\_  CALCULATED  \_HoldupCorrelation | Input.DP\_HCOR.Value= | 362 | 1. BEGGS-BRILL 2. FLANIGAN 3. EATON 4. HOOG 5. HUGH 6. SLACK 7. ORKI 8. AWR 9. LOCK-MART 10. H-BROWN 11. USER-SUBR 12. HTFS |
| **if you chose SPECIFIED:** | | | | |
| Liquid Hold up profile | BLK\_RPLUG\_Set\_  SPECIFIED  \_HoldupProfilLIQUID | Input.HOLDUP.#0.Value =  Input.HLOCK.#1.Value = 1 | 363 | 1. HoldupList 2. LocationList |
| Solid Hold up Profile | BLK\_RPLUG\_Set\_  SPECIFIED  \_HoldupProfilSOLID | Input.SHLOC.#0.Value = 1  Input.SHOLDUP.#0.Value | 364 | 1. HoldupList 2. LocationList |

## Catalysts Input Page 7:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Activate Catalyst | BLK\_RPLUG\_Set  \_CatalystPresent | Input.CAT\_PRESENT.VALUE | 371 | 1. YES 2. NO |
| **If NO was chosen: no inputs are needed** | | | | |
| **If YES was chosen:** | | | | |
| Ignore Catalyst Volume? | BLK\_RPLUG\_Set\_  IgnoreCatalystVolume | Input.IGN\_CAT\_VOL.Value | 372 | 1. YES 2. NO |
| Catalyst weight loaded | BLK\_RPLUG\_Set\_  WeightOfCatalystLoaded | Input.CATWT.Value | 373 |  |
| Particle density | BLK\_RPLUG\_Set\_  ParticleDensity | Input.CAT\_RHO | 374 |  |
| Bed Void fraction | BLK\_RPLUG\_Set\_  BedVoidage | Input.BED\_VOIDAGE | 375 |  |

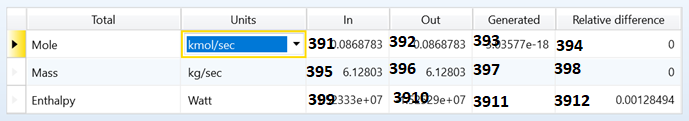
# RPLUG OUTPUT

## Summary: Page 1



|  |  |  |  |
| --- | --- | --- | --- |
| **RPlug Outputs Page 1: Summary** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Heat duty: | BLK\_RPLUG\_Get\_Heatduty | Output.QCALC.Value | 381 |
| Minimum Reactor temperature: | BLK\_RPLUG\_Get\_  MinimumReactorTemperature | Output.TMIN.Value | 382 |
| Maximum Reactor temperature: | BLK\_RPLUG\_Get\_  MaximumReactorTemperature | Output.TMAX.Value | 383 |
| Residence time: | BLK\_RPLUG\_Get\_ResidenceTime | Output.RES\_TIME.Value | 384 |
| Thermal fluid Inlet temperature: | BLK\_RPLUG\_Get\_  ThermalFluidInletTemperature | Output.COOLANT\_TIN.Value | 385 |
| Thermal fluid Inlet Vapor fraction: | BLK\_RPLUG\_Get\_  ThermalFluidInletVaporFraction | Output.COOLANT\_VIN.Value | 386 |

## Balance: Page 2



|  |  |  |  |
| --- | --- | --- | --- |
| **RPlug Outputs Page 2: Balance** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Mole Balance: IN | BLK\_RPLUG\_Get  \_MoleFlowBalanceIN() | Output.BAL\_MOLI\_TFL.Value | 391 |
| Mole Balance: OUT | BLK\_ RPLUG\_Get  \_MoleFlowBalanceOUT() | Output.BAL\_MOLO\_TFL.Value | 392 |
| Mole Balance: Relative difference | BLK\_ RPLUG\_Get  \_MoleFlowBalanceRelDifference() | Output.BAL\_MOLR\_TFL.Value | 393 |
| Mole Balance: Generated | BLK\_ RPLUG\_Get  \_MoleFlowBalanceGenerated | Outpu. BAL\_MOLG\_TFL.Value | 394 |
| Mass Balance: IN | BLK\_ RPLUG\_Get  \_MassFlowBalanceIN | Output.BAL\_MASI\_TFL.Value | 395 |
| Mass Balance: OUT | BLK\_ RPLUG\_Get  \_MassFlowBalanceOUT | Output.BAL\_MASO\_TFL.Value | 396 |
| Mass Balance: Relative difference | BLK\_ RPLUG\_Get  \_MassFlowBalanceRelDifference | Output.BAL\_MASR\_TFL.Value | 397 |
| Mass Balance: Generated | BLK\_ RPLUG\_Get  \_MassFlowBalanceGenerated | Output. BAL\_MASG\_TFL.Value | 398 |
| Enthalpy: IN | BLK\_ RPLUG\_Get  \_EnthalpyBalanceIN | Output.TOT\_ENTH\_ABS.Value | 399 |
| Enthalpy: OUT | BLK\_ RPLUG\_Get  \_EnthalpyBalanceOUT | Output.BAL\_ENTH\_OUT.Value | 3910 |
| Enthalpy: Relative difference | BLK\_ RPLUG\_Get  \_EnthalpyBalanceRelDifference | Output.TOT\_ENTH\_REL.Value | 3911 |
| Enthalpy: Generated | BLK\_ RPLUG\_Get  \_EnthalpyBalanceGenerated | Output. BAL\_ENTH\_GEN.Value | 3912 |

## Distribution: Page 3

Graphical user interface, application

Description automatically generated

Not implemented

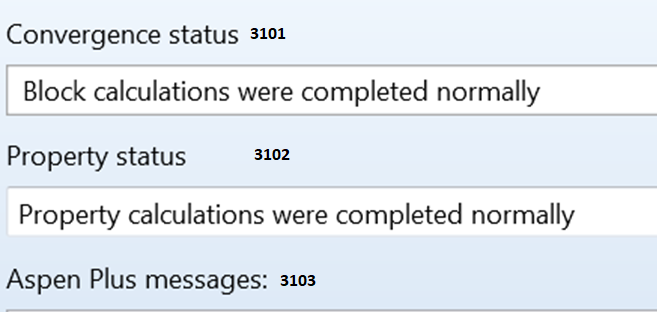
## Polymer Attributes: Page 4

Table

Description automatically generated

Not implemented

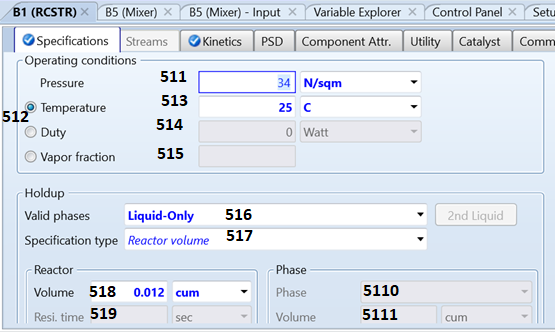
## Status: Page 5



|  |  |  |  |
| --- | --- | --- | --- |
| **RPlug Outputs Page 5: Status** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Convergence status: | BLK\_RPLUG\_Get\_ConvergenceStatus() | Output.BLKSTAT.Value = 0 | 3101 |
| Convergence Message: | BLK\_RPLUG\_Get\_ConvergenceMessage() | Output.BLKMSG.Value | 3102 |
| Property status: | BLK\_RPLUG\_Get\_PropertyStatus() | Output.PROPSTAT.Value = 0 | 3103 |

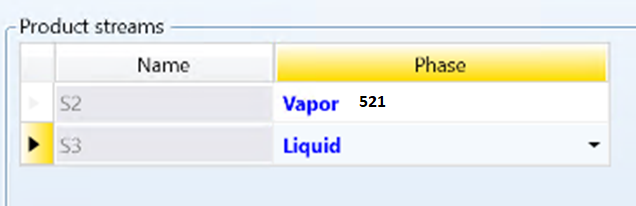
# RCSTR Inputs

## Specifications Page 1:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Pressure | BLK\_CISTR\_Set\_Pressure | Input.PRES.Value | 511 |  |
| Input Option: |  | Input.SPEC\_OPT | 512 | 1. TEMP 2. DUTY 3. VFRAC |
| **If TEMP was chosen:** | | | | |
|  | BLK\_CISTR\_Set\_Temperature | Input.TEMP.Value | 513 |  |
| **If DUTY was chosen:** | | | | |
| Duty | BLK\_CISTR\_Set\_Duty | Input.DUTY.Value | 514 |  |
| **If VFRAC was chosen:** | | | | |
| Vapor fraction | BLK\_CISTR\_Set\_VaporFraction | Input.VFRAC.Value | 515 |  |
| Valid Phases: | BLK\_CISTR\_Set\_Phases | Input.NPHASE.Value  Input.PHASE.Value | 516 | Phase number   1. 1 2. 2 3. 3   Phase   1. V 2. L |
| Specification type | BLK\_CISTR\_Set\_Specification\_type | Input.SPEC\_TYPE  .Value | 517 | 1. TOT-VOL 2. RES-TIME 3. TOT-VOL-PH-VOL 4. TOT-VOL-PH-VOL-FRAC 5. TOT-VOL-PH-RES-TIME 6. RES-TIME-PH-VOL-FRAC 7. PH-RES-TIME-PH-VOL-FRAC |
| **If TOT-VOL chosen:** | | | | |
| Volume: | BLK\_CISTR\_Set\_Volume | Input.VOL.Value | 518 |  |
| **If RES-TIME was chosen:** | | | | |
| Residence time: | BLK\_CISTR\_Set\_  ResidenceTime | Input.RES\_TIME  .Value | 519 |  |
| **If TOT-VOL-PH-VOL was chosen:** | | | | |
| Total volume | BLK\_CISTR\_Set\_Volume | Input.VOL.Value | 518 |  |
| Phase that is held up | BLK\_CISTR\_Set\_Specification\_PhaseHoldup | Input.SPEC\_PHASE | 5110 | 1. VAPOR 2. CONDENSED |
| Volume of the phase specified for Phase Holdup: | BLK\_CISTR\_Set\_Volume\_of\_PhaseHoldup | Input.REACT\_VOL | 5111 |  |
| **If TOT-VOL-PH-VOL-FRAC was chosen:** | | | | |
| Total volume | BLK\_CISTR\_Set\_Volume | Input.VOL.Value | 518 |  |
| Phase that is held up | BLK\_CISTR\_Set\_Specification\_PhaseHoldup | Input.SPEC\_PHASE | 5110 | 1. VAPOR 2. CONDENSED |
| Volume fraction of the phase specified for Phase Holdup | BLK\_CISTR\_Set\_VolumeFrac\_of\_PhaseHoldup | Input.REACT\_VOL\_FR | 5112 |  |
| **If TOT-VOL-PH-RES-TIME was chosen:** | | | | |
| Total volume | BLK\_CISTR\_Set\_Volume | Input.VOL.Value | 518 |  |
| Phase that is held up | BLK\_CISTR\_Set\_Specification\_PhaseHoldup | Input.SPEC\_PHASE | 5110 | 1. VAPOR 2. CONDENSED |
| Residence time specified for Phase Holdup: | BLK\_CISTR\_Set\_  Residencetime\_of  \_PhaseHoldup | Input.PH\_RES\_TIME | 5113 |  |
| **If RES-TIME-PH-VOL-FRAC was chosen:** | | | | |
| Residence time: |  | Input.RES\_TIME | 519 |  |
| Phase that is held up | BLK\_CISTR\_Set\_Specification\_PhaseHoldup | Input.SPEC\_PHASE | 5110 | 1. VAPOR 2. CONDENSED |
| Volume of the phase specified for Phase Holdup: | BLK\_CISTR\_Set\_Volume\_of\_PhaseHoldup | Input.REACT\_VOL | 5111 |  |
| **If PH-RES-TIME-PH-VOL-FRAC was chosen:** | | | | |
| Phase that is held up | BLK\_CISTR\_Set\_Specification\_PhaseHoldup | Input.SPEC\_PHASE | 5110 | 1. VAPOR 2. CONDENSED |
| Volume fraction of the phase specified for Phase Holdup: | BLK\_CISTR\_Set\_VolumeFrac\_of\_PhaseHoldup | Input.REACT\_  VOL\_FR | 5112 |  |
| Residence time specified for Phase Holdup: | BLK\_CISTR\_Set\_Residencetime\_of\_PhaseHoldup | Input.PH\_RES\_TIME | 5113 |  |

## Streams Page 2:



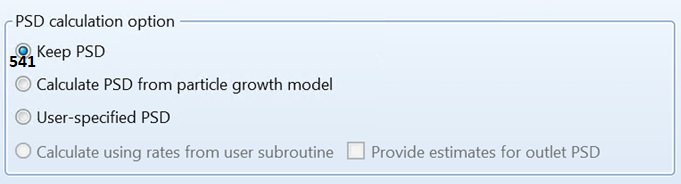
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Product stream  phase | BLK\_CISTR\_Set\_  Productstream\_phase | Input.PROD\_PHASE.  ”STREAMNAME”.Value | 521 | Streamname  Possible Values:   1. V 2. L 3. L1 4. L2 5. W 6. VL 7. VL1 8. LW 9. L1L2 |

## Kinetics Page 3:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Activate Reactive System | BLK\_CISTR\_Set\_  ActivateReactions | Input.REACSYS.Value | 531 | 1. YES 2. NO |
| Select Reaction Set: |  |  | 532 |  |
| Add activities: | BLK\_CISTR\_Set  \_ReactionActivities | Input.ACT\_VALUE.#0.Value | 533 | 1. ActivityList 2. ActivityNameList |
| Activate Crystallization System: | BLK\_CISTR\_Set\_Activate\_Crystalization | Input.CRYSTSYS.Value | 534 | 1. YES 2. NO |
| Include Agitation: | BLK\_CISTR\_Set\_Activate\_Agitation | Input.AGITATOR.Value  Input.AGITRATE.Value  Input.IMPELLR\_DIAM.Value  Input.POWERNUMBER.Value | 535 | Rotationrate, ImpellerDiameter, Powernumber,  ActivateAgitation\_or\_not:   1. ”NO” 2. ”YES” |

## Particle Size Determination PSD Page 4:



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Function description** | **Function name** | | **Variable path** | | **Function number** | | **Extra Inputs/Options** |
| Calculation Options: | BLK\_CISTR\_Set\_Calculation\_Option | | Input.OPT\_PSD | | 541 | | 1. COPY 2. CONSTANT 3. SPEC |
| **If you chose COPY: no input needed** | | | | | | | |
| **If you chose CONSTANT:** | | | | | | | |
| Growth model option | BLK\_CISTR\_Set\_  ParticalGrowthModel | Input.  CONST\_METHOD  .Value | | 542 | | A) “DELTAD-NUM“  Shift number distribution by same delta D  B) “DELTAD-MASS“  C) “DELTAV-NUM“  Shift number distribution by same delta V  D) “EQUI-MASS“  Mass distribution equivalent growth  E)“EQUI-SURFACE“  Surface distribution equivalent growth  F)“EQUI-NUMBER“  Number distribution equivalent growth | |
| **If SPEC was chosen: not implemented yet** | | | | | | | |

## Component Attributes Page 5

Graphical user interface, application

Description automatically generated

Not implemented

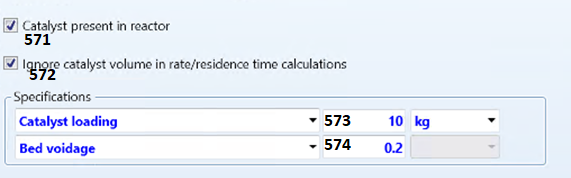
## Utilities Page 6:

Graphical user interface, application

Description automatically generated with medium confidence

Not implemented

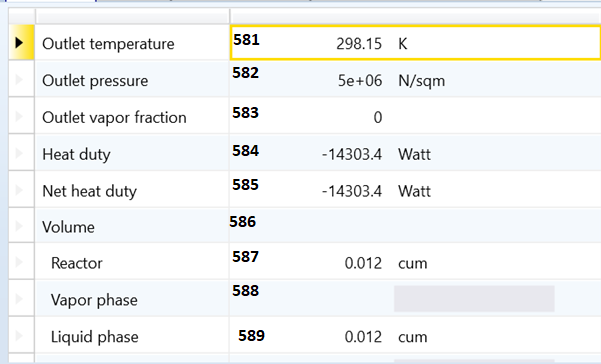
## Catalyst Page 7:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Activate Catalyst | BLK\_CISTR\_Set  \_CatalystPresent | Input.CAT\_PRESENT.VALUE | 571 | 1. YES 2. NO |
| **If NO was chosen: no inputs are needed** | | | | |
| **If YES was chosen:** | | | | |
| Ignore Catalyst Volume? | BLK\_CISTR\_Set\_  IgnoreCatalystVolume | Input.IGN\_CAT\_VOL.Value | 572 | 1. YES 2. NO |
| Catalyst weight loaded | BLK\_CISTR\_Set\_  WeightOfCatalystLoaded | Input.CATWT.Value | 573 |  |
| Particle density | BLK\_CISTR\_Set\_  ParticleDensity | Input.CAT\_RHO | 574 |  |

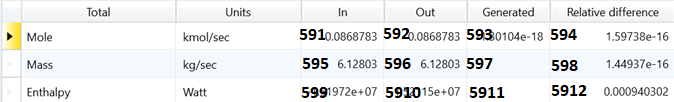
# RCSTR Output:

## Summary Page 1:



|  |  |  |  |
| --- | --- | --- | --- |
| **RCSTR Outputs Page 1: Summary** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Outlet temperature: | LK\_RCSTR\_Get\_OutletTemperature | Output.B\_TEMP.Value | 581 |
| Outlet pressure: | BLK\_RCSTR\_Get\_OutletPressure | Output.B\_PRES.Value | 582 |
| Outlet vapor fraction: | BLK\_RCSTR\_Get\_OutletVaporFraction | Output.B\_VFRAC.Value | 583 |
| Heat duty: | BLK\_RCSTR\_Get\_HeatDuty | Output.QCALC.Value | 584 |
| Net heat duty: | BLK\_RCSTR\_Get\_NetHeatDuty | Output.QNET.Value | 585 |
| Reactor Volume: | BLK\_RCSTR\_Get\_ReactorVolume | Output.TOT\_VOL.Value | 586 |
| Vapor phase Volume: | BLK\_RCSTR\_Get\_VaporPhaseVolume | Output.VAP\_VOL.Value | 587 |
| Liquid phase Volume: | BLK\_RCSTR\_Get\_LiquidPhaseVolume | Output.LIQ\_VOL.Value | 588 |
| Liquid 1 phase volume: | BLK\_RCSTR\_Get\_Liquid1PhaseVolume | Output.LIQ1\_VOL | 589 |
| Salt phase volume: | BLK\_RCSTR\_Get\_SaltPhaseVolume | Output.SALT\_VOL | 5810 |
| Condensed phase volume: | BLK\_RCSTR\_Get  \_CondensedPhaseVolume | Output.COND\_VOL | 5811 |
| Reactor residence time: | BLK\_RCSTR\_Get  \_ReactorResidenceTime | Output.TOT\_RES\_TIME | 5812 |
| Vapor phase residence time: | BLK\_RCSTR\_Get  \_VaporPhaseResidenceTime | Output.VAP\_RES\_TIME | 5813 |
| Condensed phase residence time: | BLK\_RCSTR\_Get  \_CondensedPhaseResidenceTime | Output.COND\_RES\_TIM | 5814 |

## Balance Page 2:



|  |  |  |  |
| --- | --- | --- | --- |
| **RCSTR Outputs Page 2: Balance** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Mole Balance: IN | BLK\_RCSTR\_Get  \_MoleFlowBalanceIN() | Output.BAL\_MOLI\_TFL.Value | 591 |
| Mole Balance: OUT | BLK\_RCSTR \_Get  \_MoleFlowBalanceOUT() | Output.BAL\_MOLO\_TFL.Value | 592 |
| Mole Balance: Relative difference | BLK\_ RCSTR \_Get  \_MoleFlowBalanceRelDifference() | Output.BAL\_MOLR\_TFL.Value | 593 |
| Mass Balance: IN | BLK\_ RCSTR\_Get  \_MassFlowBalanceIN | Output.BAL\_MASI\_TFL.Value | 594 |
| Mass Balance: OUT | BLK\_ RCSTR\_Get  \_MassFlowBalanceOUT | Output.BAL\_MASO\_TFL.Value | 595 |
| Mass Balance: Relative difference | BLK\_ RCSTR\_Get  \_MassFlowBalanceRelDifference | Output.BAL\_MASR\_TFL.Value | 596 |
| Enthalpy: IN | BLK\_ RCSTR\_Get  \_EnthalpyBalanceIN | Output.TOT\_ENTH\_ABS.Value | 597 |
| Enthalpy: OUT | BLK\_ RCSTR\_Get  \_EnthalpyBalanceOUT | Output.BAL\_ENTH\_OUT.Value | 598 |
| Enthalpy: Relative difference | BLK\_ RCSTR\_Get  \_EnthalpyBalanceRelDifference | Output.TOT\_ENTH\_REL.Value | 599 |

## Reaction Kinetics Page 3:

Graphical user interface, application, table

Description automatically generated

Not implemented

None of them can be extracted. It just gives an error “This function is not allowed for this variable.”

## Component Generation Rates Page 4:

Graphical user interface, application, table

Description automatically generated

Not implemented

Same error as Page 3

None of them can be extracted. It just gives an error “This function is not allowed for this variable.”

## Custom Reaction Variables Page 5:

Not implemented

## Utility Usage Page 6:

Not implemented

## Distributions Page 7:

Not implemented

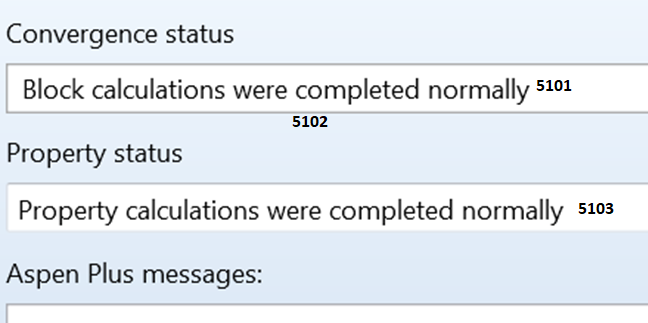
## Polymer Attributes Page 8:

Not implemented

## Crystallization Page 9:

Not implemented

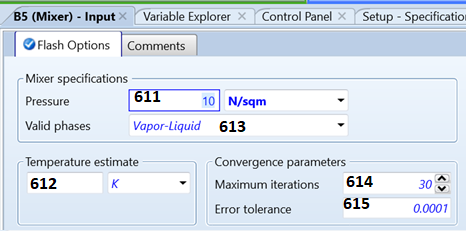
## Status Page 10:



|  |  |  |  |
| --- | --- | --- | --- |
| **RCSTR Outputs Page 10: Status** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Convergence status: | BLK\_RCSTR\_Get  \_ConvergenceStatus | Output.BLKSTAT.Value = 0 | 5101 |
| Convergence Message: | BLK\_RCSTR\_Get  \_ConvergenceMessage | Output.BLKMSG.Value | 5102 |
| Property status: | BLK\_RCSTR\_Get  \_PropertyStatus | Output.PROPSTAT.Value = 0 | 5103 |

# Mixer Input

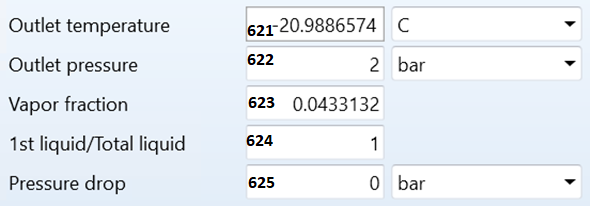
## Flash Option Page 1:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/ Options** |
| Pressure: | BLK\_MIXER\_Set\_Pressure | Input.PRES.Value | 611 |  |
| Temperature Estimate | BLK\_MIXER\_Set\_  TemperatureEstimate | Input.T\_EST.Value | 612 |  |
| Number of Phases and Phase: | BLK\_MIXER\_Set\_Phases | Input.NPhase.Value  Input.Phase.Value | 613 |  |
| Maximum iterations: | BLK\_MIXER\_Set\_  MaximumIteration | Input.MAXIT.Value | 614 |  |
| Error tolerance: | BLK\_MIXER\_Set\_  ErrorTolerance | Input.TOL.Value | 615 |  |

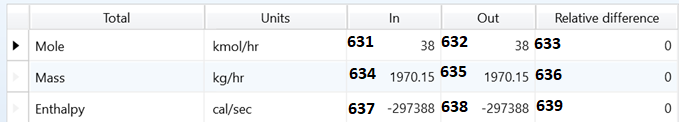
# MIXER Output

## Summary Page 1:



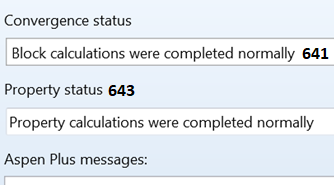
|  |  |  |  |
| --- | --- | --- | --- |
| **Mixer Outputs Page 1: Summary** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Outlet temperature | BLK\_MIXER\_Get\_OutletTemperature | Output.B\_TEMP | 621 |
| Outlet pressure | BLK\_MIXER\_Get\_OutletPressure | Outlet.B\_PRES | 622 |
| Vapor fraction | BLK\_MIXER\_Get\_VaporFraction | Outlet.B\_VFRAC | 623 |
| First liquid by total liquid | BLK\_MIXER\_Get\_FirstLiquidbyTotalLiquid | Outlet.LIQ\_RATIO | 624 |
| Pressure drop | BLK\_MIXER\_Get\_PressureDrop | Outlet.PDROP | 625 |

## Balance Page 2:



|  |  |  |  |
| --- | --- | --- | --- |
| **Mixer Outputs Page 2: Balance** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Mole Balance: IN | BLK\_MIXER\_Get\_MoleFlowBalanceIN | Output.BAL\_MOLI\_TFL | 631 |
| Mole Balance: OUT | BLK\_MIXER\_Get\_  MoleFlowBalanceOUT | Output.BAL\_MOLO\_TFL | 632 |
| Mole Balance: Relative difference | BLK\_MIXER\_Get\_  MoleFlowBalanceRelDifference | Output.BAL\_MOLR\_TFL | 633 |
| Mass Balance: IN | BLK\_MIXER\_Get\_  MassFlowBalanceIN | Output.BAL\_MASI\_TFL | 634 |
| Mass Balance: OUT | BLK\_MIXER\_Get\_  MassFlowBalanceOUT | Output.BAL\_MASO\_TFL | 635 |
| Mass Balance: Relative difference | BLK\_MIXER\_Get\_  MassFlowBalanceRelDifference | Output.BAL\_MASR\_TFL | 636 |
| Enthalpy: IN | BLK\_MIXER\_Get\_EnthalpyBalanceIN | Output.TOT\_ENTH\_ABS | 637 |
| Enthalpy: OUT | BLK\_MIXER\_Get\_EnthalpyBalanceOUT | Output.BAL\_ENTH\_OUT | 638 |
| Enthalpy: Relative difference | BLK\_MIXER\_Get\_  EnthalpyBalanceRelDifference | Output.TOT\_ENTH\_REL | 639 |

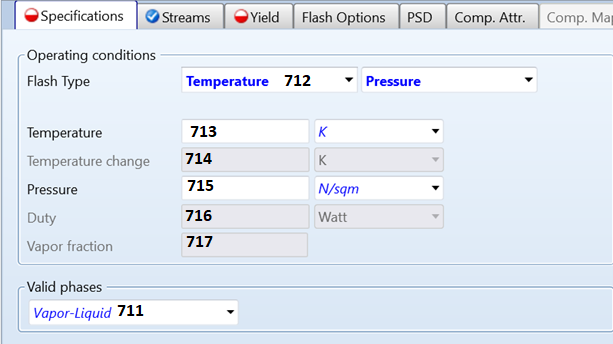
## Status Page 3:



|  |  |  |  |
| --- | --- | --- | --- |
| **Mixer Outputs Page 7: Status** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Convergence status: | BLK\_MIXER\_Get\_ConvergenceStatus() | Output.BLKSTAT.Value = 0 | 641 |
| Convergence Message: | BLK\_MIXER\_Get\_ConvergenceMessage() | Output.BLKMSG.Value | 642 |
| Property status: | BLK\_MIXER\_Get\_PropertyStatus() | Output.PROPSTAT.Value = 0 | 643 |

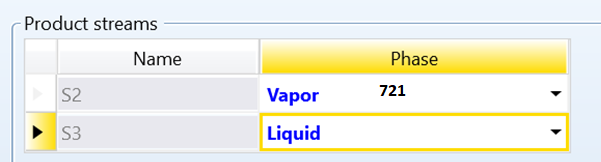
# RYIELD Input

## Specifications Page 1



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Valid phase | BLK\_RYIELD\_Set\_Phases | Input.PHASE.Value  Input.NPHASE.Value | 711 |  |
| Flash Type Option | BLK\_RYIELD\_Set\_  FlashTypeOption | Input.SPEC\_OPT | 712 | 1. TP 2. TV 3. PV 4. TD 5. PD 6. PQ ? 7. TQ ? 8. DTP 9. DTV 10. DTD 11. DTQ ? |
| **For the given Input choice, the following inputs can be set:** | | | | |
| Temperature  T | BLK\_RYIELD\_Set\_Temperature | Input.TEMP.Value | 713 |  |
| Pressure  P | BLK\_RYIELD\_Set\_Pressure | Input.PRES.Value | 714 |  |
| Temperature change  DT | BLK\_RYIELD\_Set\_  TemperatureChange | Input.DELT.Value | 715 |  |
| Duty  D | BLK\_RYIELD\_Set\_Duty | Input.DUTY.Value | 716 |  |
| Vapor fraction  V | BLK\_RYIELD\_Set\_  Vaporfraction | Input.VFRAC.Value | 717 |  |

## Streams Page 2:

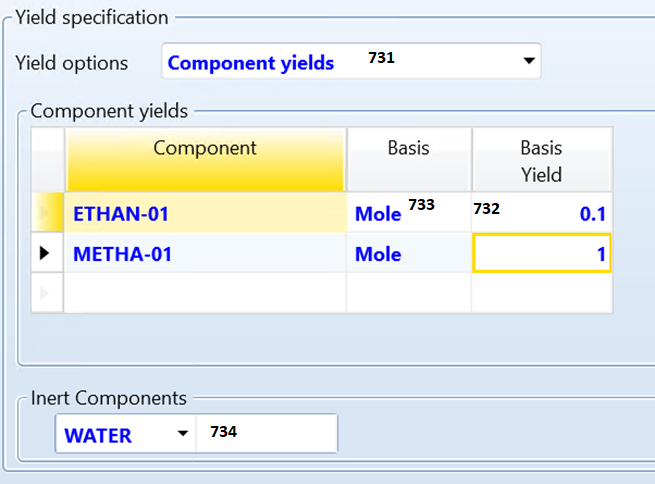


This is only necessary if you have two Output streams:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Product stream  phase | BLK\_RYIELD\_Set\_  PhaseOfProductStream | Input.PROD\_PHASE.  ”STREAMNAME”.Value | 721 | Streamname  Possible Values:   1. V 2. L 3. L1 4. L2 5. W 6. VL 7. VL1 8. LW 9. L1L2 |

## Yield Page 3:

A) If Component yields were selected:



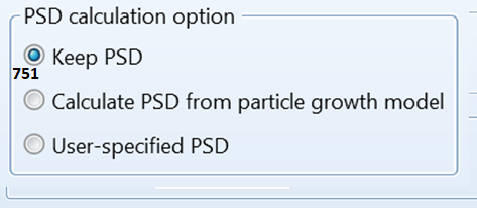
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Yield options: | BLK\_RYIELD\_Set\_  YieldCalcOption | Input.USER\_YIELD.Value | 731 | 1. Component yields “NO“ 2. User Subroutine “YES“ 3. Component mapping “NO2“ 4. Petro characterization “NO3“ |
| **If Component yield was chosen:** | | | | |
| Setting the Yield per Basis flow: | BLK\_RYIELD\_Set\_  ComponentYield\_  YieldPerFlow | Input.MOLE\_YIELD.”ETHAN-01 MIXED”.Value | 732 |  |
| Changing the Basis: | BLK\_RYIELD\_Set  \_ComponentYield\_  ChangeBasis | Input.BASIS.”METHA-01 MIXED”.Value | 733 | 1. “MASS” 2. “MOLE” |
| Setting Inert Components: | BLK\_RYIELD\_Set\_  ComponentYield\_  InertComponent | Input.COMP\_LIST.#0.Value = WATER | 734 |  |
| **If User Subroutine was selected: not implemented yet** | | | | |
| **If Component mapping was selected: not implemented yet** | | | | |
| **If Petro characterization was selected: not implemented yet** | | | | |

## Flash Options Page 4:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Estimation of the Pressure: | BLK\_RYIELD\_Set\_Temperature  Estimation | Input.PRES.Value | 741 |  |
| Temperature Estimate | BLK\_RYIELD\_Set\_Pressure  Estimation | Input.T\_EST.Value | 742 |  |
| Maximum iterations: | BLK\_RYIELD\_Set\_  MaximumIteration | Input.MAXIT.Value | 743 |  |
| Error tolerance: | BLK\_RYIELD\_Set\_  ErrorTolerance | Input.TOL.Value | 744 |  |

## Particle Size Determination PSD Page 5:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Calculation Options: | BLK\_RYIELD\_Set\_  Calculation\_Option | Input.OPT\_PSD | 751 | 1. COPY 2. CONSTANT 3. SPEC |
| **If you chose COPY: no input needed** | | | | |
| **If you chose CONSTANT:** | | | | |
| Growth model option | BLK\_RYIELD\_Set\_  ParticalGrowthModel | Input.CONST\_METHOD  .Value | 752 | A) “DELTAD-NUM“  Shift number distribution by same delta D  B) “DELTAD-MASS“  C) “DELTAV-NUM“  Shift number distribution by same delta V  D) “EQUI-MASS“  Mass distribution equivalent growth  E)“EQUI-SURFACE“  Surface distribution equivalent growth  F)“EQUI-NUMBER“  Number distribution equivalent growth |
| **If SPEC was chosen: not clear** | | | | |

## Component Attribute Page 6

missing

## Component mapping Page 7

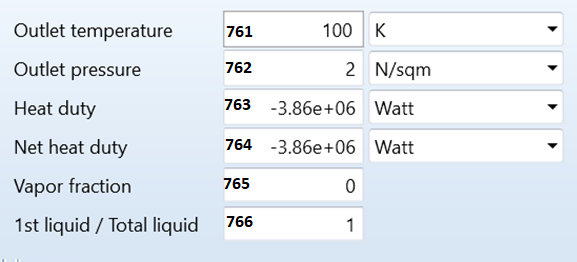
Graphical user interface

Description automatically generated

missing

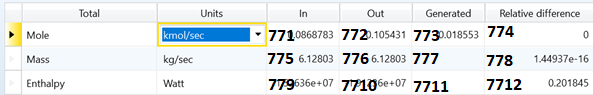
# RYIELD Output:

## Summary Page 1:



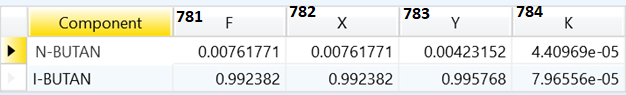
|  |  |  |  |
| --- | --- | --- | --- |
| **RYield Outputs Page 1: Summary** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Outlet temperature: | BLK\_RYIELD\_Get\_OutletTemperature | Output.B\_TEMP | 761 |
| Outlet pressure: | BLK\_RYIELD\_Get\_OutletPressure | Output.B\_PRES | 762 |
| Heat duty: | BLK\_RYIELD\_Get\_HeatDuty | Output.QCALC | 763 |
| Net heat duty: | BLK\_RYIELD\_Get\_NetHeatDuty | Output.QNET | 764 |
| Vapor fraction: | BLK\_RYIELD\_Get\_VaporFraction | Output.B\_VFRAC | 765 |
| 1st liquid to total liquid fraction: | BLK\_RYIELD\_Get\_  FirstLiquidbyTotalLiquidFraction | Output.LIQ\_RATIO | 766 |

## Balance Page 2:



|  |  |  |  |
| --- | --- | --- | --- |
| **RYield Outputs Page 2: Balance** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Mole Balance: IN | BLK\_RYIELD\_Get  \_MoleFlowBalanceIN() | Output.BAL\_MOLI\_TFL.Value | 771 |
| Mole Balance: OUT | BLK\_RYIELD\_Get  \_MoleFlowBalanceOUT() | Output.BAL\_MOLO\_TFL.Value | 772 |
| Mole Balance: Relative difference | BLK\_RYIELD\_Get  \_MoleFlowBalanceRelDifference() | Output.BAL\_MOLR\_TFL.Value | 773 |
| Mole Balance: Generated | BLK\_RYIELD\_Get  \_MoleFlowBalanceGenerated | Outpu. BAL\_MOLG\_TFL.Value | 774 |
| Mass Balance: IN | BLK\_RYIELD\_Get  \_MassFlowBalanceIN | Output.BAL\_MASI\_TFL.Value | 775 |
| Mass Balance: OUT | BLK\_RYIELD\_Get  \_MassFlowBalanceOUT | Output.BAL\_MASO\_TFL.Value | 776 |
| Mass Balance: Relative difference | BLK\_RYIELD\_Get  \_MassFlowBalanceRelDifference | Output.BAL\_MASR\_TFL.Value | 777 |
| Mass Balance: Generated | BLK\_RYIELD\_Get  \_MassFlowBalanceGenerated | Output. BAL\_MASG\_TFL.Value | 778 |
| Enthalpy: IN | BLK\_RYIELD\_Get  \_EnthalpyBalanceIN | Output.TOT\_ENTH\_ABS.Value | 779 |
| Enthalpy: OUT | BLK\_RYIELD\_Get  \_EnthalpyBalanceOUT | Output.BAL\_ENTH\_OUT.Value | 7710 |
| Enthalpy: Relative difference | BLK\_RYIELD\_Get  \_EnthalpyBalanceRelDifference | Output.TOT\_ENTH\_REL.Value | 7711 |
| Enthalpy: Generated | BLK\_RYIELD\_Get  \_EnthalpyBalanceGenerated | Output. BAL\_ENTH\_GEN.Value | 7712 |

## Phase Equilibrium Page 3:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RYield Outputs Page 3: Phase Equilibrium** | | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** | **Inputs** |
| Total Flow fraction | BLK\_RYIELD\_Get\_  TotalFlowFraction() | Output.F.”Compoundname”  .Value | 781 | Compound name |
| Liquid concentration: | BLK\_RYIELD\_Get\_  Liquidconcentrations() | Output.X.”Compoundname”  .Value | 782 | Compound name |
| Vapor concentration: | BLK\_RYIELD\_Get\_  Vaporconcentrations() | Output.Y.”Compoundname”  .Value | 783 | Compound name |
| Equilibrium constant: | BLK\_RYIELD\_Get\_  EquilibriumConstant() | Output.B\_K.”Compoundname”  .Value | 784 | Compound name |

## Weight distribution Page 4:

Table

Description automatically generated

Not implemented

## Pseudocomp Breakdown Page 5:

Graphical user interface, application

Description automatically generated

Not implemented

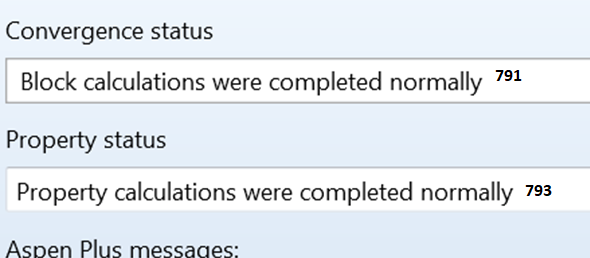
## Utility usage Page 6:

Application

Description automatically generated with medium confidence

Not implemented

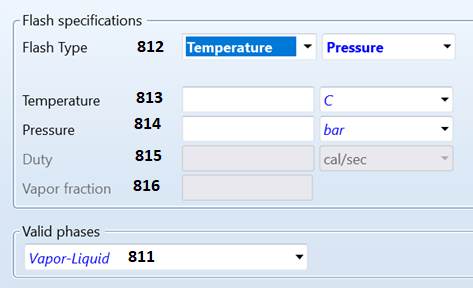
## Status Page 7:



|  |  |  |  |
| --- | --- | --- | --- |
| **RYield Outputs Page 5: Status** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Convergence status: | BLK\_RYIELD\_Get\_ConvergenceStatus() | Output.BLKSTAT.Value = 0 | 791 |
| Convergence Message: | BLK\_RYIELD\_Get\_ConvergenceMessage() | Output.BLKMSG.Value | 792 |
| Property status: | BLK\_RYIELD\_Get\_PropertyStatus() | Output.PROPSTAT.Value = 0 | 793 |

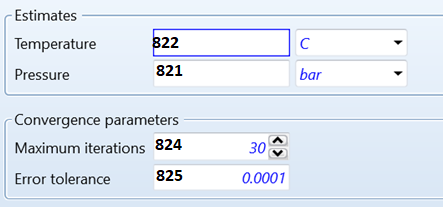
# FLASH2 Input

## Specifications Page 1:



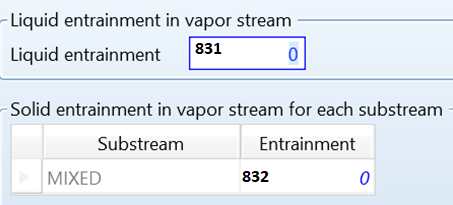
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | | **Function number** | **Extra Inputs/Options** |
| Valid phase | BLK\_FLASH2\_Set\_Phases | Input.PHASE.Value  Input.NPHASE.Value | | 811 |  |
| Flash Type Option | BLK\_FLASH2\_Set\_  FlashTypeOption | Input.SPEC\_OPT | | 812 | 1. TP 2. TV 3. PV 4. TD 5. PD 6. PQ ? 7. TQ ? |
| **For the given Input choice the following inputs can be set:** | | | | | |
| Temperature  T | BLK\_FLASH2\_Set\_Temperature | | Input.TEMP.Value | 813 |  |
| Pressure  P | BLK\_FLASH2\_Set\_Pressure | | Input.PRES.Value | 814 |  |
| Duty  D | BLK\_FLASH2\_Set\_Duty | | Input.DUTY.Value | 815 |  |
| Vapor fraction  V | BLK\_FLASH2\_Set\_Vapor\_fraction | | Input.VFRAC.Value | 816 |  |

## Flash Option Page 2:



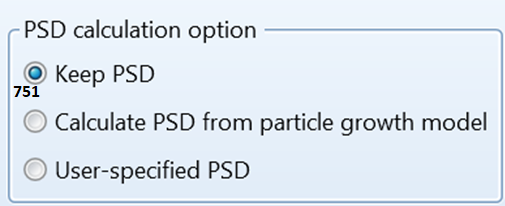
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Pressure: | BLK\_FLASH2\_Set\_  PressureEstimation | Input.PRES.Value | 821 |  |
| Temperature Estimate | BLK\_FLASH2\_Set\_  TemperatureEstimation | Input.T\_EST.Value | 822 |  |
| Maximum iterations: | BLK\_FLASH2\_Set\_  MaximumIteration | Input.MAXIT.Value | 824 |  |
| Error tolerance: | BLK\_FLASH2\_Set\_  ErrorTolerance | Input.TOL.Value | 825 |  |

## Entrainment Page 3:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Liquid Entrainment | BLK\_FLASH2\_Set\_  PressureEstimation | Input.ENTRN.Value | 831 |  |
| Solid Entrainment | BLK\_FLASH2\_Set\_  TemperatureEstimation | Input.VAPOR.MIXED.Value | 832 |  |

## Particle Size Determination (PSD) Page 4:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Calculation Options: | BLK\_FLASH2\_Set\_  Calculation\_Option | Input.OPT\_PSD | 841 | 1. COPY 2. CONSTANT 3. SPEC |
| **If you chose COPY: no input needed** | | | | |
| **If you chose CONSTANT:** | | | | |
| Growth model option | BLK\_FLASH2\_Set\_  ParticalGrowthModel | Input.CONST\_METHOD  .Value | 842 | A) “DELTAD-NUM“  Shift number distribution by same delta D  B) “DELTAD-MASS“  C) “DELTAV-NUM“  Shift number distribution by same delta V  D) “EQUI-MASS“  Mass distribution equivalent growth  E)“EQUI-SURFACE“  Surface distribution equivalent growth  F)“EQUI-NUMBER“  Number distribution equivalent growth |
| **If SPEC was chosen: not clear** | | | | |

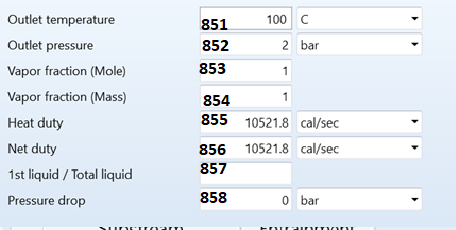
## Utilities Page 5:

Graphical user interface, application

Description automatically generated

# FLASH2 Output:

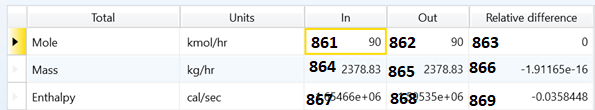
## Summary PAGE 1



|  |  |  |  |
| --- | --- | --- | --- |
| **Flash2 Outputs Page 1: Summary** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Outlet temperature: | BLK\_FLASH2\_Get\_OutletTemperature | Output.B\_TEMP | 851 |
| Outlet pressure: | BLK\_FLASH2\_Get\_OutletPressure | Output.B\_PRES | 852 |
| Vapor fraction (mole): | BLK\_FLASH2\_Get\_VaporFractionMole | Output.B\_VFRAC | 853 |
| Vapor fraction (mass): | BLK\_FLASH2\_Get\_VaporFractionMass | Output.MVFRAC | 854 |
| Heating duty: | BLK\_FLASH2\_Get\_HeatingDuty | Output.QCALC | 855 |
| Net duty: | BLK\_FLASH2\_Get\_NetDuty | Output.QNET | 856 |
| 1st liquid/Total liquid: | LK\_FLASH2\_Get\_  FirstLiquidtoTotalLiquidRatio | Output.LIQ\_RATIO | 857 |
| Pressure drop: | BLK\_FLASH2\_Get\_PressureDrop | Output.PDROP | 858 |

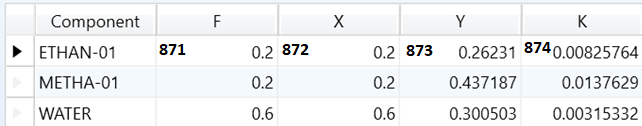
## Balance PAGE 2:

Same as the other Balances



|  |  |  |  |
| --- | --- | --- | --- |
| **Flash2 Outputs Page 2: Balance** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Mole Balance: IN | BLK\_FLASH2\_Get\_MoleFlowBalanceIN() | Output.BAL\_MOLI\_TFL.Value | 861 |
| Mole Balance: OUT | BLK\_FLASH2\_Get\_  MoleFlowBalanceOUT() | Output.BAL\_MOLO\_TFL.Value | 862 |
| Mole Balance: Relative difference | BLK\_FLASH2\_Get\_  MoleFlowBalanceRelDifference() | Output.BAL\_MOLR\_TFL.Value | 863 |
| Mass Balance: IN | BLK\_FLASH2\_Get\_  MassFlowBalanceIN() | Output.BAL\_MASI\_TFL.Value | 864 |
| Mass Balance: OUT | BLK\_FLASH2\_Get\_  MassFlowBalanceOUT() | Output.BAL\_MASO\_TFL.Value | 865 |
| Mass Balance: Relative difference | BLK\_FLASH2\_Get\_  MassFlowBalanceRelDifference() | Output.BAL\_MASR\_TFL.Value | 866 |
| Enthalpy: IN | BLK\_FLASH2\_Get\_EnthalpyBalanceIN() | Output.TOT\_ENTH\_ABS.Value | 867 |
| Enthalpy: OUT | BLK\_FLASH2\_Get\_  EnthalpyBalanceOUT() | Output.BAL\_ENTH\_OUT.Value | 868 |
| Enthalpy: Relative difference | BLK\_FLASH2\_Get\_  EnthalpyBalanceRelDifference() | Output.TOT\_ENTH\_REL.Value | 869 |

## Phase Equilibrium PAGE 3:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Flash2 Outputs Page 3: Phase Equilibrium** | | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** | **Inputs** |
| Total Flow fraction | BLK\_FLASH2\_Get\_  TotalFlowFraction\_F | Output.F.”Compoundname”  .Value | 871 | Compound name |
| Liquid concentration: | BLK\_FLASH2\_Get\_  LiquidConcentration\_X | Output.X.”Compoundname”  .Value | 872 | Compound name |
| Vapor concentration: | BLK\_FLASH2\_Get\_  VaporConcentration\_Y | Output.Y.”Compoundname”  .Value | 873 | Compound name |
| Equilibrium constant: | BLK\_FLASH2\_Get\_  EquilinriumConstant\_K | Output.B\_K.”Compoundname”  .Value | 874 | Compound name |

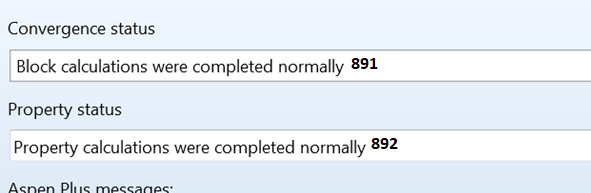
## Utility Usage PAGE 4:

Application

Description automatically generated

Not implemented

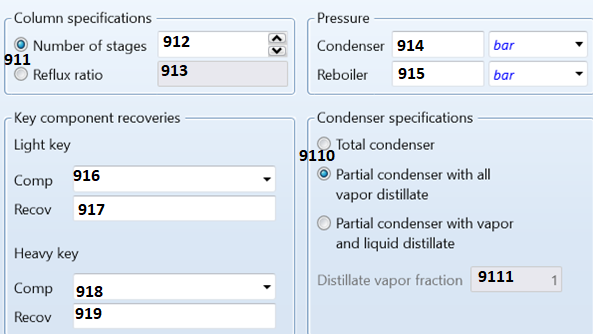
## Status PAGE 5:



|  |  |  |  |
| --- | --- | --- | --- |
| **Flash2 Outputs Page 5: Status** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Convergence status: | BLK\_FLASH2\_Get\_ConvergenceStatus | Output.BLKSTAT.Value = 0 | 891 |
| Convergence Message: | BLK\_FLASH2\_Get\_ConvergenceMessage | Output.BLKMSG.Value | 892 |
| Property status: | BLK\_FLASH2\_Get\_PropertyStatus | Output.PROPSTAT.Value = 0 | 893 |

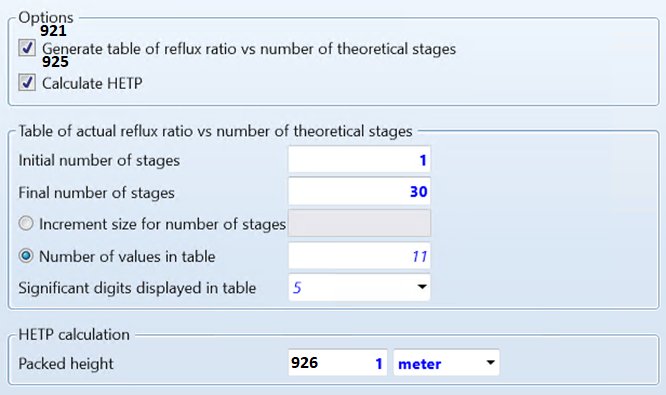
# DSTWU Input

## Specification PAGE 1:



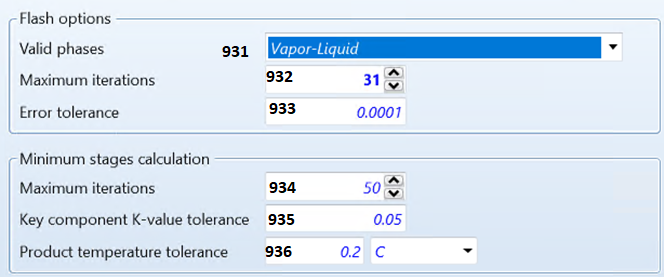
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Specification Option: | BLK\_DSTWU\_Set\_  StageRefluxOption | Input.OPT\_NTRR | 911 | 1. NSTAGE 2. RR |
| **If you chose NSTAGE:** | | | | |
| Number of Stages | BLK\_DSTWU\_Set\_  NumberOfStages | Input.NSTAGE | 912 |  |
| **If you chose RR:** | | | | |
| Reboiler ratio | BLK\_DSTWU\_Set\_  Refluxratio | Input.RR | 913 |  |
| Condenser Pressure: | BLK\_DSTWU\_Set\_  CondenserPressure | Input.PTOP | 914 |  |
| Reboiler Pressure: | BLK\_DSTWU\_Set\_  ReboilerPressure | Input.PBOT | 915 |  |
| Lightkey Component: | BLK\_DSTWU\_Set\_  LightkeyComponent | Input.LIGHTKEY | 916 |  |
| Lightkey Recovery: | BLK\_DSTWU\_Set\_  HeavykeyComponent | Input.RECOVL | 917 |  |
| Heavykey Component: | BLK\_DSTWU\_Set\_  LightkeyRecovery | Input.HEAVYKEY | 918 |  |
| Heavykey Recovery: | BLK\_DSTWU\_Set\_  HeavykeyRecovery | Input.RECOVH | 919 |  |
| Condenser specification: | BLK\_DSTWU\_Set\_  CondenserOption | Input.OPT\_RDV | 9110 | 1. LIQUID 2. VAPOR 3. VAPLIQ |
| **If VAPLIQ was chosen:** | | | | |
| Distillate vapor fraction: | BLK\_DSTWU\_Set\_  VAPLIQ\_  DestillVaporFraction | Input.RDV | 9111 |  |

## Calculation Options PAGE 2:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Generate Table of reflux Option | BLK\_DSTWU\_Set\_  GenerateTableOption | Input.PLOT. | 921 | 1. YES 2. NO |
| **If YES was chosen:** | | | | |
| First stage for calculation: | BLK\_DSTWU\_Set\_  GenerateTable\_FirstStage | Input.LOWER | 922 |  |
| Last stage for calculation: | BLK\_DSTWU\_Set\_  GenerateTable\_LastStage | Input.UPPER | 923 |  |
| Number of Stages in calculation: | BLK\_DSTWU\_Set\_  GenerateTable\_StageNumber | Input.NPOINT | 924 |  |
| Calculate HETP Option | BLK\_DSTWU\_Set\_  CalculateHeightequivalent  HETP\_Option | Input.OPT\_CALHETP | 925 | 1. YES 2. NO |
| **If YES was chosen:** | | | | |
| Packed Height: | BLK\_DSTWU\_Set\_  CalculateHeightequivalent  HETP\_PackedHeight | Input.PACK\_HEIGHT | 926 |  |

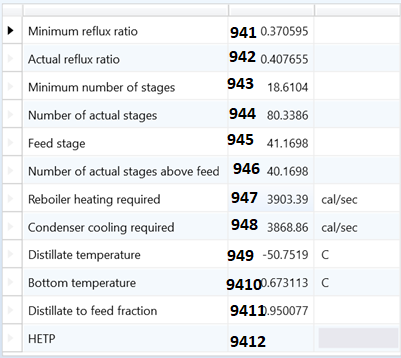
## Convergence PAGE 3:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Free water option as included in phase definition: | BLK\_DSTWU\_Set\_  FreewaterOption | Input.BLKOPFREWAT | 931 | 1. YES 2. NO 3. DIRTY |
| Max number of Flash iterations | BLK\_DSTWU\_Set\_  MaxNumber  FlashIterations | Input.FLASH\_MAXIT | 932 |  |
| Flash convergence tolerance | BLK\_DSTWU\_Set\_  FlashConvergence  Tolerance | Input.FLASH\_TOL | 933 |  |
| Max number of min. Stages calculation: | BLK\_DSTWU\_Set\_  MaxNumber  MinStageIterations | Input.MAXIT | 934 |  |
| Tolerance on K value for minimum stages: | BLK\_DSTWU\_Set\_  KvalueTolerance | Input.K\_TOL  Input.TEMP\_TOL | 935 |  |
| Tolerance on product temperature: | BLK\_DSTWU\_Set\_  ProductTemp  Tolerance | Input.TEMP\_TOL | 936 |  |

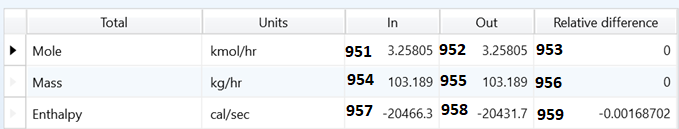
# DSTWU Outputs:

## Summary PAGE 1:



|  |  |  |  |
| --- | --- | --- | --- |
| **DSTWU Outputs Page 1: Summary** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Minimum reflux ratio: | BLK\_DSTWU\_Get\_MinimumRefluxRatio | Output.MIN\_REFLUX.Value | 941 |
| Actual reflux ratio: | BLK\_DSTWU\_Get\_ActualRefluxRatio | Output.ACT\_REFLUX.Value | 942 |
| Minimum number of stages: | BLK\_DSTWU\_Get\_MinimumNStage | Output.MIN\_STAGES.Value | 943 |
| Actual number of stages: | BLK\_DSTWU\_Get\_ActualNStage | Output.ACT\_STAGES.Value | 944 |
| Actual number of stages above feed: | BLK\_DSTWU\_Get\_ActualNumber  OfStagesAboveFeed | Output.FEED\_LOCATN.Value | 945 |
| Feed stage: | BLK\_DSTWU\_Get\_FeedStage | Output.FEED\_LOCATN.Value | 946 |
| Reboiler heating required: | BLK\_DSTWU\_Get\_ReboilerHeatingRequired | Output.RECT\_STAGE.Value | 947 |
| Condenser cooling required: | BLK\_DSTWU\_Get\_CondenserCoolingRequired | Output.COND\_DUTY.Value | 948 |
| Destillate temperature: | BLK\_DSTWU\_Get\_DestillateTemperature | Output.DISTIL\_TEMP.Value | 949 |
| Bottom temperature: | BLK\_DSTWU\_Get\_BottomTemperature | Output.BOTTOM\_TEMP.Value | 9410 |
| Destillate to feed fraction: | BLK\_DSTWU\_Get\_DestillateFeedFraction | Output.DIST\_VS\_FEED.Value | 9411 |
| HETP: | BLK\_DSTWU\_Get\_HETP | Output.HETP.Value | 9412 |

## Balance PAGE 2:



|  |  |  |  |
| --- | --- | --- | --- |
| **DSTWU Outputs Page 2: Balance** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Mole Balance: IN | BLK\_DSTWU\_Get\_MoleFlowBalanceIN() | Output.BAL\_MOLI\_TFL.Value | 951 |
| Mole Balance: OUT | BLK\_ DSTWU\_Get\_  MoleFlowBalanceOUT() | Output.BAL\_MOLO\_TFL.Value | 952 |
| Mole Balance: Relative difference | BLK\_ DSTWU\_Get\_  MoleFlowBalanceRelDifference() | Output.BAL\_MOLR\_TFL.Value | 953 |
| Mass Balance: IN | BLK\_ DSTWU\_Get\_  MassFlowBalanceIN() | Output.BAL\_MASI\_TFL.Value | 954 |
| Mass Balance: OUT | BLK\_ DSTWU\_Get\_  MassFlowBalanceOUT() | Output.BAL\_MASO\_TFL.Value | 955 |
| Mass Balance: Relative difference | BLK\_ DSTWU\_Get\_  MassFlowBalanceRelDifference() | Output.BAL\_MASR\_TFL.Value | 956 |
| Enthalpy: IN | BLK\_ DSTWU\_Get\_EnthalpyBalanceIN() | Output.TOT\_ENTH\_ABS.Value | 957 |
| Enthalpy: OUT | BLK\_DSTWU\_Get\_EnthalpyBalanceOUT() | Output.BAL\_ENTH\_OUT.Value | 958 |
| Enthalpy: Relative difference | BLK\_ DSTWU\_Get\_  EnthalpyBalanceRelDifference() | Output.TOT\_ENTH\_REL.Value | 959 |

## Reflux ratio profile PAGE 3:

The reflux profile is defined by the inputs: They are not sorted by the numbering of the values but rather by the Stage number. This makes extracting the data more complicated and it is necessary to loop through all the “Stagenumbers” and then extract the values afterwards to assign them together.

“Blockname”.Output.RR\_Out.”Stagenumber”.Value

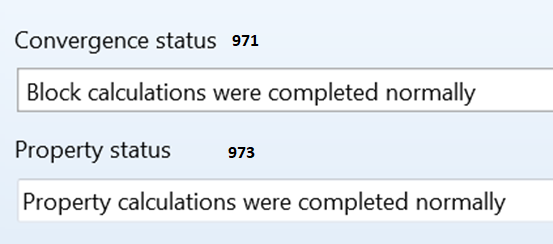
Table

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Reflux ratio profile: | BLK\_DSTWU\_Get\_RefluxRatioProfile | Output.RR\_Out.”Stagenumber”.Value | 961 |

Careful! This one return two lists where one contains the stage numbers and the second one contains the reflux ratio at each of the stages.

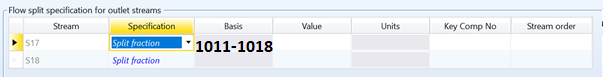
## 21.4. Status PAGE 4:



|  |  |  |  |
| --- | --- | --- | --- |
| **DSTWU Outputs Page 4: Status** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Convergence status: | BLK\_DSTWU\_Get\_ConvergenceStatus | Output.BLKSTAT.Value = 0 | 971 |
| Convergence Message: | BLK\_DSTWU\_Get\_ConvergenceMessage | Output.BLKMSG.Value | 972 |
| Property status: | BLK\_DSTWU\_Get\_PropertyStatus | Output.PROPSTAT.Value = 0 | 973 |

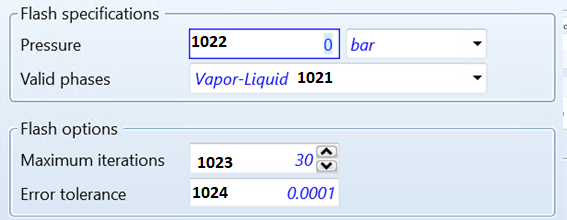
# FSPLITTER Input

## Specification PAGE 1:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs/Options** |
| Split fraction: | BLK\_SPLITTER\_Set  \_By\_Flow | Input.FRAC.  ”Streamname”.Value | 1011 |  |
| Flow: | BLK\_SPLITTER\_Set  \_By\_Flow | Input.BASIS\_FLOW.  ”Streamname.Value” | 1012 |  |
| Actual volume flow: | BLK\_SPLITTER\_Set  \_By\_ActualVolumeFlow | Input.VOL\_FLOW.  ”Streamname”.Value | 1013 |  |
| Limit flow: | BLK\_SPLITTER\_Set  \_By\_LimitFlow | Input.BASIS\_LIMIT.  ”Streamname”.Value | 1014 |  |
| Volume limit flow: | BLK\_SPLITTER\_Set  \_By\_VolumeLimitFlow | Input.VOL\_LIMIT.  ”Streamname”.Value | 1015 |  |
| Cum limit flow: | BLK\_SPLITTER\_Set  \_By\_CumLimitFlow | Input.BASIS\_C\_LIM.  ”Streamname”.Value | 1016 |  |
| Cum volume limit flow: | BLK\_SPLITTER\_Set  \_By\_CumVolumeLimitFlow | Input.VOL\_C\_LIM.  ”Streamname”.Value | 1017 |  |
| Residual fraction: | BLK\_SPLITTER\_Set  \_By\_ResidualFraction | Input.R\_FRAC.  ”Streamname”.Value | 1018 |  |

## Flash Options PAGE 2:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Phase number and Phase | BLK\_SPLITTER\_  Set\_Phases | Input.NPHASE1.Value  Input.PHASE1.Value | 1021 | = 1, 2, 3  = “V”, “L”, “S” |
| Estimated Pressure | BLK\_SPLITTER\_  Set\_Pressure | Input.P\_EST.Value | 1022 |  |
| Maximum iterations | BLK\_SPLITTER\_Set\_  MaximumIteration | Input.MAXIT.Value | 1023 |  |
| Error tolerance | BLK\_SPLITTER\_Set\_  ErrorTolerance | Input.TOL.Value | 1024 |  |

## Key Components PAGE 3

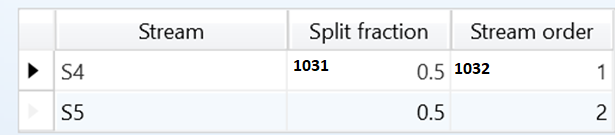
Graphical user interface

Description automatically generated

missing

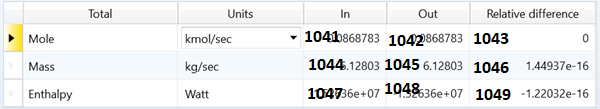
# FSPLITTER Output

## Summary PAGE 1:



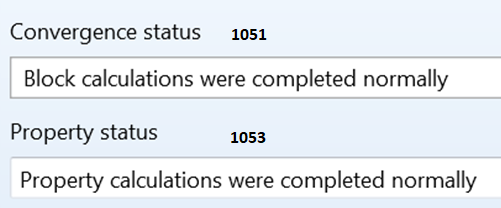
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FSplit Outputs Page 1: Summary** | | | |  |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** | **Input** |
| Splitfraction per Stream | BLK\_FSPLITTER\_Get  \_SplitFraction | Output.STREMFRAC.  Streamname.Value | 1031 | Stream name |
| Stream order per Stream | BLK\_FSPLITTER\_Get  \_StreamOrder | Output.STREAM\_ORDER.  Streamname.Value | 1032 | Stream name |

## Balances PAGE 2:



|  |  |  |  |
| --- | --- | --- | --- |
| **FSplit Outputs Page 2: Balance** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Mole Balance: IN | BLK\_FSPLITTER\_Get\_  MoleFlowBalanceIN | Output.BAL\_MOLI\_TFL.Value | 1041 |
| Mole Balance: OUT | BLK\_FSPLITTER\_Get\_  MoleFlowBalanceOUT | Output.BAL\_MOLO\_TFL.Value | 1042 |
| Mole Balance: Relative difference | BLK\_FSPLITTER\_Get\_  MoleFlowBalanceRelDifference | Output.BAL\_MOLR\_TFL.Value | 1043 |
| Mass Balance: IN | BLK\_FSPLITTER\_Get\_  MassFlowBalanceIN | Output.BAL\_MASI\_TFL.Value | 1044 |
| Mass Balance: OUT | BLK\_FSPLITTER\_Get\_  MassFlowBalanceOUT | Output.BAL\_MASO\_TFL.Value | 1045 |
| Mass Balance: Relative difference | BLK\_FSPLITTER\_Get\_  MassFlowBalanceRelDifference | Output.BAL\_MASR\_TFL.Value | 1046 |
| Enthalpy: IN | BLK\_FSPLITTER\_Get\_  EnthalpyBalanceIN | Output.TOT\_ENTH\_ABS.Value | 1047 |
| Enthalpy: OUT | BLK\_FSPLITTER\_Get\_  EnthalpyBalanceOUT | Output.BAL\_ENTH\_OUT.Value | 1048 |
| Enthalpy: Relative difference | BLK\_FSPLITTER\_Get\_  EnthalpyBalanceRelDifference | Output.TOT\_ENTH\_REL.Value | 1049 |

## Status PAGE 3:



|  |  |  |  |
| --- | --- | --- | --- |
| **FSplit Outputs Page 3: Status** | | | |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** |
| Convergence status: | BLK\_FSPLITTER\_Get\_  ConvergenceStatus | Output.BLKSTAT.Value = 0 | 1051 |
| Convergence Message: | BLK\_FSPLITTER\_Get\_  ConvergenceMessage | Output.BLKMSG.Value | 1052 |
| Property status: | BLK\_FSPLITTER\_Get\_  PropertyStatus | Output.PROPSTAT.Value = 0 | 1053 |

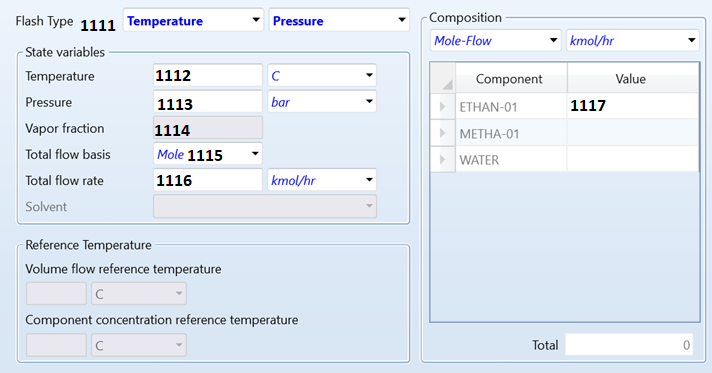
# STREAM INPUT

## Specifications PAGE 1

For all streams the general Path towards the variables which are below is:

Data.Streams.”Streamname”.Input.

This will be ignored in the next few ones because it simplifies the pathway and heightens the readability.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Flash type Option: | STRM\_Set\_FlashTypeOption | Input.MIXED\_SPEC.  MIXED.Value | 1111 | 1. TP 2. TV 3. PV |
| Temperature: | STRM\_Set\_Temperature | Input.TEMP.Value | 1112 |  |
| Pressure: | STRM\_Set\_Pressure | Input.PRES.Value | 1113 |  |
| Vapor fraction: | STRM\_Set\_VaporFraction | Input.VFRAC | 1114 |  |
| Total flow basis: | STRM\_Set\_TotalFlowRate | Input.FLOWBASE.  MIXED.Value | 1115 | A) MASS  B ) MOLE  C ) STDVOL  D ) VOLUME |
| Total flow rate: | STRM\_Set\_TotalFlowBasis | Input.TOTFLOW.Value | 1116 |  |
| Composition: (specified for each compound) | STRM\_Set\_ComponentFlowRate | Input.FLOW.MIXED.  ”Compoundname”  .Value | 1117 |  |

For all streams the general path towards the variables which are below is:

Data.Streams.”Streamname”.Input.

This will be ignored in the next few ones because it simplifies the pathway and heightens the readability.

## CI Solid PAGE 2

Graphical user interface

Description automatically generated with medium confidence

Not implemented

## NC Solid PAGE 3

Graphical user interface, table

Description automatically generated

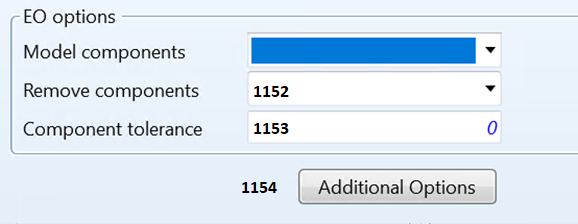
Not implemented

## Flash Options PAGE 4



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Calculate flash Calculation Option: | STRM\_Set\_  CalculateStream  PropertiesOption | Input.FL\_OPTION.  MIXED.Value | 1141 | 1. “NOFLASH” 2. “” |
| Valid phases | STRM\_Set\_Phases | Input.PHASE.MIXED.Value  Input.NPHASE.MIXED.Value | 1142 |  |
| Max number of Flash iterations | STRM\_Set\_  MaximumIteration | Input.MAXIT.MIXED | 1143 |  |
| Flash convergence tolerance: | STRM\_Set\_  ErrorTolerance | Input.TOL.MIXED | 1144 |  |

## EO Options PAGE 5

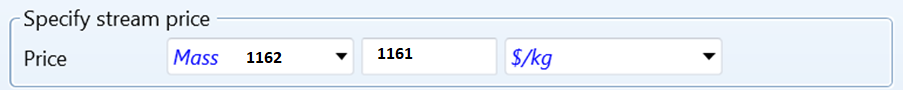


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Model component: | **Not implemented** | | | |
| Remove component: | STRM\_Set\_  RemoveComponentOption | Input.AUTO\_COMPS  .Value | 1152 | A ) ALWAYS  B ) IF-NO-COMPS  C ) NEVER |
| Component tolerance: | STRM\_Set\_  ComponentTolerance | Input.AUTO\_COMPS\_T  .Value | 1153 |  |
| Choosing Additional Options: | STRM\_Set\_  ChooseAdditionalOptions | Input.EO\_COMPS  .Value | 1154 |  |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Solution method: | STRM\_Set\_  AddOptSolution  Method | Input.SOL  \_METHOD.Value | 1155 | A ) OPEN-PERT-IN  B ) OPEN-PERT-WA  C ) OPEN-NOPERT  D ) PERTUBATION  E ) DO-NOT-CREAT |
| Open derivatives method: | STRM\_Set\_  AddOptOpen  DerivationMethod | Input.DERIV  \_METHOD.Value | 1156 | A ) ANALYTICAL  B ) NUMERICAL  C ) UPDATE-ANALY  D ) UPDATE-NUMER |
| Pass-through: | STRM\_Set\_  AddOptPassThrough | Input.PASS\_  THROUGH.Value | 1157 | 1. **YES** 2. **NO** |
| Negative comp. check: | STRM\_Set\_  AddOptNegative  ComponentCheckTol | Input.NEG\_  COMP\_CHK.Value | 1158 |  |
| Negative flow check: | STRM\_Set\_  AddOptNegative  FlowCheckTol | Input.NEG\_  FLOW\_CHK.Value | 1159 |  |
| Always Instantiate: | STRM\_Set\_  AddOptAlways  Instantiate | Input.ALWAYS\_  INST.Value | 11510 |  |
| **Flash Component Option:** | | | | |
| Sparcity: | STRM\_Set\_  AddOptSparcity | Input.SPARSITY  .Value | 11511 | A ) COMP-INDEP  B ) FULL  C ) WATER-SOLU |
| Sparcity component | Not implemented | | | |
| Lightkey: | STRM\_Set\_  AddOptLightkey | Input.EO\_LIGHT  \_KEY.Value | 11512 |  |
| Heavykey: | STRM\_Set\_  AddOptHeavykey | Input.EO\_HEAVY  \_KEY.Value | 11513 |  |
| Water only check: | STRM\_Set\_  AddOptWater  OnlyCheck | Input.CHECK\_FREE  \_W.Value | 11514 |  |
| **Flash Options** | | | | |
| Remove missing phase: | STRM\_Set\_  AddOptRemove  MissingPhase | Input.AUTO\_PHASE  .Value | 11515 |  |
| Phase tolerance: | STRM\_Set\_  AddOptPhase  Tolerance | Input.AUTO\_PHASE  \_T.Value | 11516 |  |
| Flash formulation: | STRM\_Set\_  AddOptFlash  Formulation | Input.FLASH\_FORM  .Value | 11517 | A ) PML  B ) SMOOTHING |
| **Smoothing tolerances:** | | | | |
| VfracX-Tolerance | STRM\_Set\_  AddOptSmoothing  VfracXTol | Input.VFRACX\_TOL  .Value | 11518 |  |
| Vfrac-Tolerance | STRM\_Set\_  AddOptSmoothing  VfracTol | Input.VFRAC\_TOL  .Value | 11519 |  |
| Sfrac-Tolerance | STRM\_Set\_  AddOptSmoothing  SfracTol | Input.SFRAC\_TOL  .Value | 11520 |  |
| Composition Tolerance | STRM\_Set\_  AddOptSmoothing  CompositionTol | Input.COMP\_TOL  .Value | 11521 |  |
| Temperature Tolerance | STRM\_Set\_AddOpt  Smoothing  TemperatureTol | Input.EO\_TEMP\_  TOL.Value | 11522 |  |

## Costing PAGE 6



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Variable path** | **Function number** | **Extra Inputs** |
| Price per Unit: | STRM\_Set\_PricePerUnit | Input.PRICE.Value | 1161 |  |
| Change Unit: | STRM\_Set\_ChangePriceUnit | Input.PRICE.**Basis** | 1162 |  |

# STREAM OUTPUTS:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function description** | **Function name** | **Path in Aspen** | **Function number** | **Inputs** |
| Stream source: | STRM\_Get\_Source() | STRM.Streamname.Output  .STR\_MAIN.SOURCE.Value | 1171 | Stream name |
| Stream destination: | STRM\_Get\_Destination() | STRM.Streamname.Output  .STR\_MAIN.DESTINATION.Value | 1172 | Stream name |
| Stream Phase: | STRM\_Get\_Phase() | STRM.Streamname.Output  .STR\_MAIN.COMPTYPE.Value | 1173 | Stream name |
| Property set: | STRM\_Get\_PropertySet() | STRM.Streamname.Output  .STR\_MAIN.PPOPSET.Value | 1174 | Stream name |
| total mol flow per compound | STRM\_Get\_  MoleFlowPerCompound() | STRM.Streamname.Output  .STR\_MAIN.MOLEFLOW  .MIXED.Compoundname.Value | 1175 | Stream name,  Compound name |
| total mass flow per compound | STRM\_Get\_  MassFlowPerCompound() | STRM.Streamname.Output  .STR\_MAIN.MASSFLOW  .MIXED.Compoundname.Value | 1176 | Stream name,  Compound name |
| total volume flow per compound | STRM\_Get\_VolumeFlow() | STRM.Streamname.Output  .STR\_MAIN.VOLFLMX  .MIXED.Value | 1177 | Stream name |
| Mole frac by Compound: | STRM\_Get\_  MoleFracPerCompound() | STRM.Streamname.Output  .MOLEFRAC.MIXED  .Compoundname.Value | 1178 | Stream name,  Compound name |
| Mass frac by Compound: | STRM\_Get\_  MassFracPerCompound() | STRM.Streamname.Output  .MASSFRAC.MIXED  .Compoundname.Value | 1179 | Stream name,  Compound name |
| Temperature: | STRM\_Get\_Temperature() | STRM.Streamname.Output  .STR\_MAIN.TEMP.MIXED.Value | 11710 | Stream name |
| Pressure: | STRM\_Get\_Pressure() | STRM.Streamname.Output  .STR\_MAIN.PRES.MIXED.Value | 11711 | Stream name |
| Liquid concentration: | STRM\_Get\_Liquid  ConcentrationPerCompound() | STRM.Streamname.Output  .X.Compoundname.Value | 11712 | Stream name |
| Vapor concentration: | STRM\_Get\_Vapor  ConcentrationPerCompound() | STRM.Streamname.Output  .Y.Compoundname.Value | 11713 | Stream name |
| Vapor fraction: | STRM\_Get\_VaporFraction() | STRM.Streamname.Output  .STR\_MAIN.VFRAC.MIXED.Value | 11714 | Stream name |
| Liquid fraction | STRM\_Get\_LiquidFraction() | STRM.Streamname.Output  .STR\_MAIN.LFRAC.MIXED.Value | 11715 | Stream name |
| Solid fraction: | STRM\_Get\_SolidFraction() | STRM.Streamname.Output  .STR\_MAIN.SFRAC.MIXED.Value | 11716 | Stream name |