

# POWER AND WATER CORPORATION

## ASIM REFERENCE MANUAL

Prepared for Power and Water Corporation

By Radical Systems Pty Ltd

# 1 Contents

1	Contents .....	2	8.5.1	Spinning Reserve .....	14
2	License & Warranty .....	3	8.5.2	Solar Control.....	15
3	Overview.....	3	8.5.3	Fuel Efficiency .....	15
4	Prerequisites.....	3	8.5.4	Fuel Consumption .....	17
5	File Formats .....	3	8.5.5	Redundancy exceeded alarm .....	17
6	Time Formats.....	4	8.5.6	Sheddable Load .....	17
7	Analyser .....	4	8.5.7	Generator Setpoint filter .....	19
7.1	Starting.....	4	8.5.8	Service Intervals .....	21
7.2	Program Options .....	5	8.6	Generator Validation .....	22
7.3	Asim Inputs.....	9	8.7	Run Time Extensions.....	22
7.4	Analyser Templates.....	10	9	Importing data from other applications.....	24
8	Simulator .....	10	10	Modifying.....	24
8.1	Starting.....	11	10.1	License & Copyright.....	24
8.1.1	Synopsis .....	11	10.2	Obtaining the Source Code.....	24
8.1.2	Example .....	11	10.3	Submitting patches and bug fixes.....	24
8.2	Running .....	11	10.4	Editing the source code .....	24
8.3	Output Statistics.....	12	10.5	Sharing Variables .....	25
8.3.1	Automatic Statistic Generation.....	12	10.6	Implementing the IActor interface .....	25
8.3.2	Custom Statistic Generation .....	13	10.7	Order of Operations .....	26
8.4	Scaling .....	14	10.8	Performance & Speed.....	26
8.5	Operation .....	14	11	Parameter Reference .....	26

## 2 License & Warranty

THIS PROGRAM IS FREE SOFTWARE: YOU CAN REDISTRIBUTE IT AND/OR MODIFY IT UNDER THE TERMS OF THE GNU GENERAL PUBLIC LICENSE AS PUBLISHED BY THE FREE SOFTWARE FOUNDATION, EITHER VERSION 3 OF THE LICENSE, OR (AT YOUR OPTION) ANY LATER VERSION.

THIS PROGRAM IS DISTRIBUTED IN THE HOPE THAT IT WILL BE USEFUL, BUT WITHOUT ANY WARRANTY; WITHOUT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. SEE THE GNU GENERAL PUBLIC LICENSE FOR MORE DETAILS.

The GNU General Public License is located in the file COPYING.rtf in your installation folder, or it can be downloaded from <http://www.gnu.org/licenses/>.

## 3 Overview

**Asim** is a two-part tool to simulate basic diesel and renewable mini-grid installations on a one-second basis. **Asim** is comprised of the Analyser and Simulator.

The Analyser both prepares data for input to, and formats data read from the Simulator.

The Simulator is the core control-system component

The two parts communicate by using CSV files and command line arguments, so either one can be used without the other.

## 4 Prerequisites

Both parts of the **Asim** require the Microsoft .NET 4.0 runtime.

Microsoft Excel 2003 – 2013 is required to run the Analyser. Office 365 includes a subscription to the latest version of Excel, so Office 365 compatibility may change.

An Excel Macro is required to run various tasks from with Microsoft Excel. Please see the **ASIM QUICK START GUIDE** in the *docs* folder of your installation for details regarding installing this Macro.

## 5 File Formats

Input and output files are **CSV formatted**. The first row must start with the letter 't', followed by column headings. Subsequent rows start with an accepted time format, followed by values matching the header row.

The *t* column must be monotonically increasing, but successive values can be any positive offset from the previous value.

```
t, Gen1MaxP, Gen2MaxP, Gen3MaxP, Gen4MaxP
0, 80, 80, 80, 100
1, 80, 80, 100, 100
200, 80, 80, 100, 500
205, 80, 100, 500, 500
```

Figure 1 Example text file

## 6 Time Formats

Various time formats can be parsed by the **Asim**:

Human Readable Time	This includes ISO8601, and time formats like “29-Mar-2010 00:30:03.740”, “08/07/2011 19:20:30 AM”, etc.
Seconds Since the Epoch	The number of seconds since the epoch, or “Unix time” is expressed as the number of seconds since midnight on January 1, 1970, UTC.
Relative time	The number of seconds since the start of the simulation, eg. 0, 10, 20, etc.



- *Different files can contain different time formats. Eg. some relative to the start of the simulation, and some absolute*
- Only the first time stamp in the file will be used to determine the time format.
- If the first timestamp is a number representing seconds, and it is less than the Simulation start time (in seconds since the Epoch), then it is interpreted as relative to the start of the simulation.
- All times should be monotonically increasing, however duplicated or backward stepping time values are skipped over.
- Millisecond time values are rounded to the nearest second, and further values in the same second are ignored

## 7 Analyser

The typical way of running the model is through the Analyser. The Analyser is integrated into Microsoft Excel so that parameters can be configured easily.

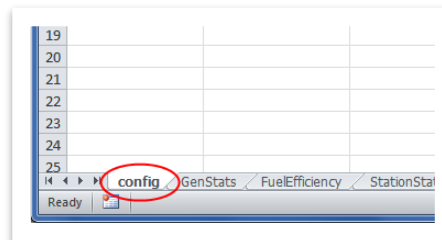
### 7.1 Starting

Start the Analyser by opening the example Microsoft Excel file Example.xlsm.<sup>1</sup>

<sup>1</sup> The Analyser uses a Visual Basic Macro to start a “Console Application” which then uses Microsoft Office Excel Primary Interop Reference to control the running Excel instance.

## 7.2 Program Options

A worksheet with the name “config” must exist that contains various parameters for locating and starting the tools.



Cell A1 of the config worksheet must contain the word “config”. The rest of column A may contain any of the following parameters. Recommended parameters are shown in **bold**:

**FlattenApplication** Location of the ExcelReader.exe tool that will read the current spreadsheet and break it up into separate csv files.

A	B
FlattenApplication	bin\ExcelReader.exe

**Simulator** Location of the **Asim** executable.

A	B
Simulator	bin\Asim.exe

**directory** Base directory used for file reading and writing. All relative paths are relative to this directory.

A	B
directory	.\

**input** File name for extra input files (eg. for files that are too large to import into Excel, for example, a large one second dataset). Multiple input directives should be used for multiple input files.

A	B
input	load data 1s.csv

**output** The file to use for writing output. The next cells must be in a particular format.

1. Column B contains the file name, relative to the directory directive
2. Column C contains the period to write to this file (s)
3. Column D and subsequent cells are glob patterns to describe what variables to write to this file. For example, “Gen[0-9]P”, or “Gen\*Cnt”.

Multiple output files may be listed in any order, with different options.

A	B	C	D	E
output	analyse.csv	86400	*Cnt	*E
output	alloutput.csv	1	*	
output	yearly.csv	31536000	Gen[1-4]*P	*E

iterations

Number of iterations to run.

A	B
iterations	345600

Start Time

The start time of the simulation. All input data before this time will be ignored.

A	B
Start Time	1/01/2012 12:00:00 AM

RunSimulator

Specifies whether to run the simulator (TRUE, T, 1, etc) or just to output the separate csv files and exit.

A	B
RunSimulator	Yes

Community Name

An optional configuration which prefixes the value in column B to each of the output files and log files.

A	B
Community Name	Daly River Solar Test

Template

The file used for analysing output. The next cells must be in a particular format.

1. The first cell contains the file name of the template file.
2. The second cell contains the name of the output file where the data will be found. (note how this is generated from an “output” option)

The template will be opened and populated with data from the output file specified, and saved as a new file.

A	B	C
Template	Analysr Layout V4.xls	analyse.csv

Log File

This file will record the variables at the start of the simulation for future reference. This can be used to recreate a previous simulation, for example if some of the settings have been changed. The cell format is as follows:

1. Column B contains the file name of the Log File.
2. Column C and subsequent cells are glob patterns to describe what variables to write to this file. For example, “Gen[0-9]P”, or “Gen\*Cnt”.

A	B	C	D
Log File	log.txt	*Set*	GenConfig*

**Watch**

The watch file will record changes to any given variable, and the simulation time that change occurred. The cell format is as follows:

1. Column B contains the file name of the watch file
2. Column C and subsequent cells are glob patterns to describe what variables to write to this file. For example, “Gen[0-9]P”, or “Gen\*Cnt”.

A	B	C
Watch	watch.txt	Gen*StartCnt

For example, the watch file could be used to print the times that a simulation blackout occurs in the file “blackout.txt” by using the following parameters:

A	B	C
Watch	blackout.txt	StatBlackCnt

**Eval**

Use Eval to specify a file containing custom blocks of code to evaluate at run-time. See [section 8.7 Run Time Extensions](#) for more information.

Column B must contain the name of the file:

A	B
Eval	code.txt

**Parameter**

Custom advanced parameters to append to the **Asim** executable. Options include:

- Algorithm – define a DLL file containing a replacement algorithm for a particular Asim control loop. See the document [ASIM CONFIGURATION GUIDE](#) in the *docs* folder of your installation for more examples.

A	B	C	D
Parameter	Algorithm	SolarController	PWC.SLMS.Algorithms.PvSimple.dll

- GeneratorStats – replace the generator simulation with a method of guessing which generators are online based on given Gen#P values.

A	B
Parameter	GeneratorStats

**Batch Command**

A command to run after the simulator and template have been run. The first option must be the executable name, and following options are passed as variables. Note that variables with spaces must be quoted here (unlike filenames elsewhere). For example:

1. Call a batch file with the name "archive.bat":

A	B
Batch Command	archive.bat

2. Call a batch file with the name "graph.bat" in another directory and pass the argument "all":

A	B	C
Batch Command	"C:\my utils\graph.bat"	All

The following environment variables are passed to all batch commands:

Environment Variable	contents
ASIM_INPUTFILES	A comma separated, unquoted list of input files, including all "input" parameters as well as excel worksheets (tab).
ASIM_OUTPUTFILES	A comma separated, unquoted list of output files.
ASIM_COMMUNITYNAME	The string given in the Community Name option.
ASIM_ITERATIONS	The number of iterations from the last simulation.
ASIM_STARTTIME	The start time used in the last iteration.
ASIM_DIRECTORY	The directory given to the last iteration.
ASIM_EXCELFILE	The full path of the Excel file being used to control the simulation.



**Report**

A text template containing tokens which should be replaced with values from the simulation.

The cell format is as follows:

- Column B contains the name of the template file containing the tokens
- Column C contains the name of the output file with the tokens replaced with values

A	B	C
Report	template.txt	AsimReport.txt

The tokens should take the form of %token%, where the token and per-cent characters will be replaced according to the following table:

Token	replacement
Any variable in the simulation	The final value of that variable at the end of the simulation
Any Environment variable from the Batch Command list above	The value of that environment variable, the same as it would be in the Batch Command
Any System environment variable	The value of that environment variable at the time of the simulation
The special token ASIM_ELAPSEDSECONDS	The number of seconds (and milliseconds in decimal notation) taken to run the inner control loop

For example, the following text:

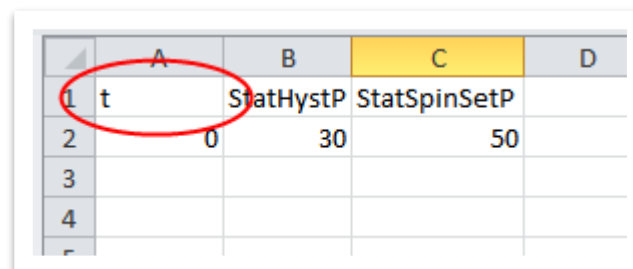
*The value of Gen1E is %Gen1E%kWh*

Would be replaced with a string like:

*The value of Gen1E is 1234.56kWh*

### 7.3 Asim Inputs

Other inputs to **Asim** can be adjusted by modifying the remaining worksheets. Each worksheet must contain the letter "t" at position A1, otherwise it will be ignored:



	A	B	C	D
1	t	StatHystP	StatSpinSetP	
2	0	30	50	
3				
4				

By setting further values for t (10, 86400, 604800, etc), the following variables can be adjusted at that time in the simulation.

## 7.4 Analyser Templates

Analysers templates are normal Microsoft Excel documents. Multiple templates may be used on each simulation run, however they take time to generate.

The Analyser will:

- Process the given template file, and populate a workbook called 'autofill' with the contents of the output csv file
- Update cell references and chart references to the autofill tab, so that formula ranges start and end at the correct cell
- Save the output under a new name with the date and time prepended to the analyser template file name.

To create / edit an analyser template, it is suggested to start with the given sample *NPV Analyser.xls*, as it contains a Helper worksheet with many predefined calculations such as:

- Simulation start time
- Simulation end time
- Generator run hours per simulated year

This Helper worksheet is not essential; however it includes many named cell references to values that can move in the Asim output files. For example, in an excel analyser template:

- Reference generator 1 run hours at the end of one year by using the autofill workbook. You will have a formula such as “=autofill!C13”
- Configure the simulator and add some output parameters to your output.csv file
- Your analyser template still references “=autofill!C13”, however generator 1 run hours may have moved to “=autofill!E25”

## 8 Simulator

- Asim is a discrete-time deterministic mini-grid simulation with many features of a power station, such as:
  - Generator Management
    - Up to 8 generators
    - Fuel and Energy statistics
    - Start / stop counters
    - Minimum run times
    - Simple on and off delays to represent warm-up, cool-down and synchronisation
    - Table of configurations specifying start order
    - Available / out of service sets
    - Hysteresis
    - Spinning Reserve
  - Renewable Energy
    - Managed solar setpoint to keep online diesel generators above minimum load (eg 40%)
    - Solar contingency coverage
  - Sheddable loads
    - Low-priority loads that can be switched off in the case of generator overload

- Sheddable loads can offset spinning reserve and solar coverage
- Black starting
  - Start a specific combination of generators (eg. all), or
  - Start only the required generators
- Extensible
  - Run-time extension language for custom functions and evaluation

## 8.1 Starting

Start **Asim** using the Analyser macro, which is initiated by selecting the “Run Application” menu item from the “Asim” menu, in the provided example spread sheet.

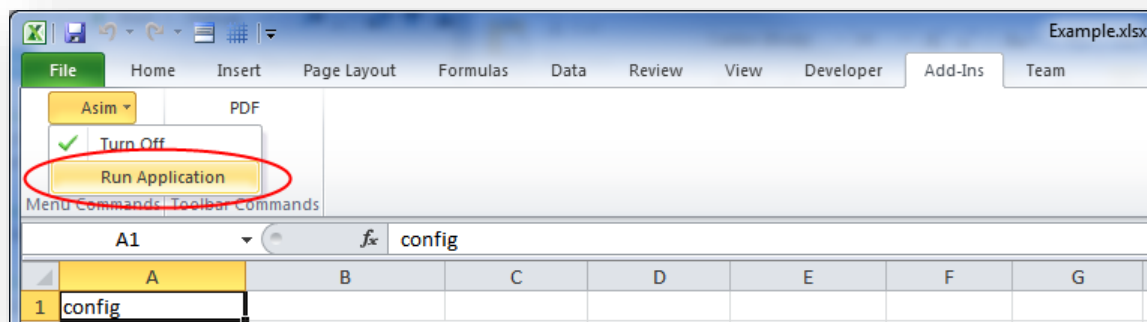


Figure 2 How to start Asim in Excel 2010

**Asim** can also be run from the command line, with options similar to those in the Analyser configuration worksheet.

### 8.1.1 Synopsis

```
Asim.exe [--iterations <iterations>] [--input <filename> [...]]
[--output [period] <varlist> [...]] [--directory <pathName>] [--StartTime
<starttime>] [--watch <watchfile> <varlist>] [--nopause]
[--algorithm <controllerName> <dllPath>] [--GeneratorStats] [--report
ReportTemplate ReportOutput]
```

The options are the same as their counterparts in the Analyser [Program Options](#) section on page 5.

--nopause tells the application not to prompt to "Press any key to continue" at the end.

All paths are system-quoted (\\ in Windows). pathName is prefixed to both input and output file names, log files and watch files.

### 8.1.2 Example

```
Asim.exe -StartTime "1/01/2012 0:00:00" --iterations 100000 --path
C:\\Users\\Joe\\Data --input config.csv --output output.csv Gen1E Gen2E
Gen3E Gen4E
```

## 8.2 Running

**Asim** will run all calculations once per iteration. Each iteration represents one second so certain counters use hard-coded values that assume this 1s frequency (energy statistics for example).

Each input file is only read up to the required line, relative to the iteration. This means that errors in input files will not be detected until the simulation reaches that iteration. Input file errors, such as incorrectly formatted numbers (e.g. "2.0f") will cause Asim to terminate with an error message noting the filename and line number that caused the error.

Each output file is written on the given period, but not flushed (forced from memory onto disk) until the simulation has finished, so files may not contain all data until the Asim process has ended.

The control system components run in the following loop:

1. Run custom evaluation blocks
2. Read all input files up to the next iteration
3. Scale values, if required (eg. for load escalation)
4. Run control system
5. Write output files, including statistics if required

### 8.3 Output Statistics

Statistics in output files can be automatically generated or customised by the user. Statistics are useful for capturing minimum and maximum values that would otherwise be lost in an average value.

If the output period is 1 second, then no statistics are generated. In this case all variables are written to the output file on each iteration, without any statistics.

#### 8.3.1 Automatic Statistic Generation

If the period is 2 seconds or more, minimum, maximum and average statistics are generated each period, and the actual variable is not output. The actual variable is replaced with five variables with the suffix:

<code>_min</code>	The minimum value of the variable since the last output
<code>_minT</code>	The occurrence of the minimum value during the last period
<code>_max</code>	The maximum value of the variable since the last output
<code>_maxT</code>	The occurrence of the maximum value during the last period
<code>_ave</code>	The average of the samples during the last period

All counters and energy totals (variables ending in Cnt or E) are monotonically increasing, therefore the minimum and maximum will always be the first and last sample in a period. For this reason, counters are always output as the value of the sample at the end of the period.

The following example shows how to configure various statistics and the respective columns that are written to the specified output file.

#### Excel Analyser configuration

#### Columns created in output file

A	B	C	D	E
output	output.csv	1	Gen1P	StatP

Gen1P, StatP

**Excel Analyser configuration**

A	B	C	D	E
output	output.csv	3600	Gen1P	StatP

**Columns created in output file**

Gen1P\_min, Gen1P\_minT, Gen1P\_max, Gen1P\_maxT, Gen1P\_ave, StatP\_min, StatP\_minT, StatP\_max, StatP\_maxT, StatP\_ave

A	B	C	D	E
output	output.csv	3600	Gen1P	StatP

Gen1StartCnt, Gen1E

A	B	C	D
output	output.csv	3600	Gen[1-4]*P

Gen1P\_min, Gen1P\_minT, Gen1P\_max, Gen1P\_maxT, Gen1P\_ave, Gen2P\_min, Gen2P\_minT, Gen2P\_max, Gen2P\_maxT, Gen2P\_ave, Gen3P\_min, Gen3P\_minT, Gen3P\_max, Gen3P\_maxT, Gen3P\_ave, Gen4P\_min, Gen4P\_minT, Gen4P\_max, Gen4P\_maxT, Gen4P\_ave

### 8.3.2 Custom Statistic Generation

Customising statistics is useful to minimise the number of columns in each output file.

To customise the statistics generated for a particular variable, you can specify a list of statistics to generate in braces (in the config worksheet). Valid options for this list are: *Min*, *MinT*, *Max*, *MaxT*, *Ave*, and *Act*. These provide the same statistics as described in *8.3.1 Automatic Statistic Generation*. The *Act* statistic will provide a “spot sample”, which is the current value at that point in time, in the same way that counters and energy totals are described above.

The following example shows how to configure custom statistics and the respective columns that are written to the specified output file.

**Excel Analyser configuration****Columns created in output file**

A	B	C	D	E
output	output.csv	1	Gen1P{Min,Max}	StatP{Ave}

Gen1P<sup>2</sup>, StatP<sup>2</sup>

A	B	C	D	E
output	output.csv	3600	Gen1P{Ave}	StatP{Act}

Gen1P\_ave, StatP

A	B	C	D	E
output	output.csv	3600	Gen1StartCnt{Ave}	Gen1E{Max}

Gen1StartCnt<sup>3</sup>, Gen1E<sup>3</sup>

<sup>2</sup> Note that statistics are not generated when the output period is 1, even if explicitly specified

<sup>3</sup> Note that statistics are not generated for counters and energy totals, regardless of output period

## 8.4 Scaling

Values can be scaled over time, so that input data can be escalated as required, such as to simulate price inflation or community load growth.

To scale a value, another parameter must be created, with the original parameter name beginning with ">". For example, to scale station load, include both "StatP" and ">StatP". StatP would contain time series data, and ">StatP" would contain the special string " $*x+k$ " where  $*$  and  $+$  are actual characters, and  $x$  and  $k$  are multiplier and offset constants, to be applied to StatP before that iteration.

To scale load values by 10% each year of simulation time, use the following values:

t	>StatP
0	*1+0
31536000	*1.10+0
63072000	*1.21+0
94608000	*1.33+0
126144000	*1.46+0

Figure 3 Sample variable scaling



- scaling input files should be specified before other input files, otherwise there may be a one-sample delay after changing the scaling factor
- Scaling is only useful for file-input data, not internally calculated data
- Note that scaling input files must be in CSV file format
- Note that the scaling factor is applied to the base StatP, not the StatP in the previous simulation period

## 8.5 Operation

### 8.5.1 Spinning Reserve

Spinning reserve is affected by various components of the system:

- Solar Coverage
- Spinning Reserve Setpoint
- Spinning Reserve method
- Sheddable Load (see section 8.5.6 *Sheddable Load* for more details)
- Generator Configurations

Spinning reserve is calculated as follows:

Solar coverage is simply a percentage of actual PV output, to cover (with unused spinning capacity) in the case of a cloud over event:

$$PvCoverage = \frac{PvP \cdot PvCvgPct}{100}$$

The actual reserve requirement is calculated from the maximum of both PvCoverage and the static StatSpinSetP

$$reserve = MAX(StatSpinSetP, PvCoverage)$$

The generator setpoint is then calculated as:

$$GenCfgSetP = LoadP - PvP + reserve$$

The generator controller will then choose a set of generators whose total summated power  $P_{[max]}$  maintains:

$$P_{[max]} > GenCfgSetP$$

Generator spinning reserve is then calculated as:

$$GenSpinP = P_{[max]} - P_{[act]}$$

### 8.5.2 Solar Control

The solar controller functions in the following order:

1. Read available solar (PvAvailP) and limit according to PvMaxLimP, if set.
2. Calculate solar setpoint (PvSetP) based on the solar setpoint method (see below)
3. Apply spinning reserve limits PvSetLimitSpinPct and PvSetLimitSpinPaPct, if set
4. Apply ramp rate limits PvSetMaxUpP and PvSetMaxDownP, if set
5. Limit solar setpoint to available solar power
6. Assume the solar array responds immediately, ie. Set PvP to PvSetP
7. Calculate spill ( $PvSpillP = PvAvailP - PvP$ )
8. Calculate energy totals PvE, PvAvailE and PvSpillE

There are various methods of calculating the solar setpoint (PvSetP) which determine the actual output of the simulated solar array (PvP). These methods can be selected by using the “Parameter Algorithm” setting in section 7.2 Program Options. They are:

Default solar controller	This method is used if no other algorithm is specified. It attempts to maintain the diesel minimum ideal loading, spinning reserve, and limits solar output to be less than total station load
Simple controller	Limits the solar setpoint to the Spinning reserve setpoint StatSpinSetP
FSC controller	Attempts to maintain the diesel minimum ideal loading while not reducing the diesel load so much that a switch down occurs
“None” controller	No control over the solar setpoint – it is always set to PvAvailP

### 8.5.3 Fuel Efficiency

Each generator uses a fuel efficiency curve to determine the fuel used (L per hour). The curve is made up of 5 points, each representing a load factor and resultant fuel consumption, with a linear function between any two points.

For example, a fuel efficiency curve may look like:

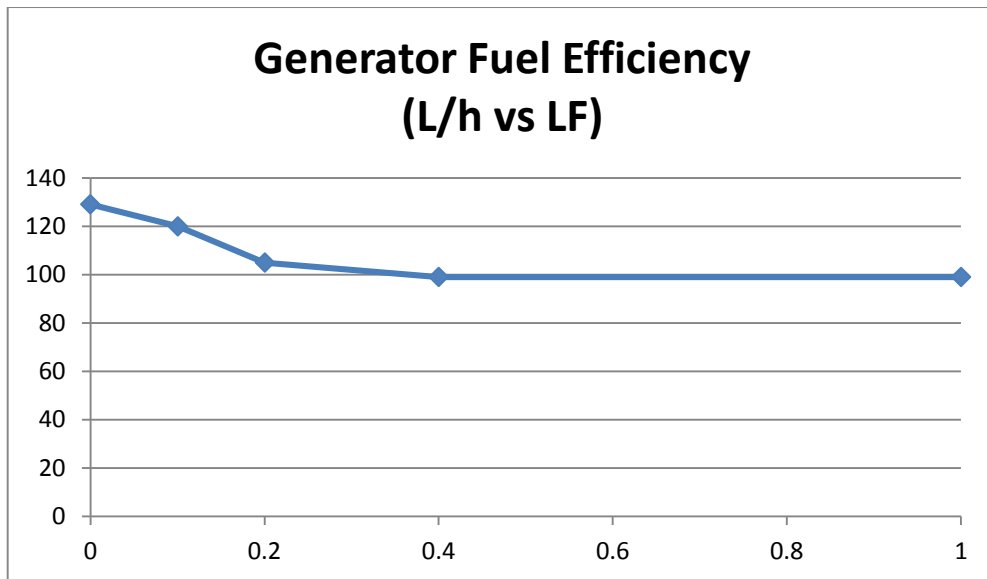


Figure 4 Sample Generator Fuel Efficiency

The x-axis shows Load Factor, where 1.0 is 100% load, and the y-axis shows fuel consumption in L/h.

This graph is made up of the points:

Point	LF	L/h
1	0	129
2	0.1	120
3	0.2	105
4	0.4	99
5	1	99

Figure 5 Sample Fuel Efficiency Points



- Fuel curve points must be monotonically increasing, or you will get “NaN” results. i.e. each LF value must be higher than the previous
- A minimum of two points must be supplied



8.5.4 Fuel Consumption

The fuel consumption  $C$ , at load factor  $L$ , between two points  $P_1[L_1, C_1]$  and  $P_2[L_2, C_2]$  is calculated as:

$$C = m(L - L_1) + C_1$$

where:

$L$  is the load factor between 0 and 1, such that:

$$L_1 \leq L < L_2$$

$m$  is the slope between the two points:

$$m = \frac{L_2 - L_1}{P_2 - P_1}$$

Total Fuel consumption  $C_T$  is calculated as:

$$C_T = C \left( \frac{1}{3600} \right) + C'$$

Where  $C'$  is the previous iterations' value for  $C_T$ ; and  $C$  is the current iterations' value for consumption. Each iteration is 1/3600<sup>th</sup> of an hour.

The following details should be considered when using Fuel Efficiency curves:

1. To use less than 5 points, disable some points by setting them to [0,0].
2. All [0,0] points must occur at the end, as any point after a [0,0] point is ignored.
3. Load Factors outside the points (eg. reverse power or overload) are calculated by projecting the nearest curve to the Load Factor value. To provide a different slope for a load factor less than zero or greater than one, you can use values for  $L$  outside of  $0 \leq L < 1$ .

8.5.5 Redundancy exceeded alarm

An "N-1" redundancy alarm can be used to determine when the capacity of all but the largest set does not satisfy the known peak load<sup>4</sup>.

For example, if 4 generators are set up in the system, if the 3 smallest sets do not cover the known peak load, then the LoadCapAl will be raised. Note: this alarm is self-resetting (it will lower or return to zero when its triggering condition is no longer active).

The Redundancy exceeded alarm *LoadCapAl* is raised when the following statement is true:

$$GenCapP < (LoadMaxP * LoadCapMargin)$$

where:

*GenCapP* is the cold reserve of the "N-1"th smallest sets:

$$GenCapP = \sum_{n=0}^{n-1} GenP_n$$

*LoadMaxP* is the maximum value for *LoadP* up to that point in the simulation

*LoadCapMargin* is the user-set safety factor to apply to *LoadMaxP*. Set *LoadCapMargin* to 1.0 to trigger the *LoadCapAl* when  $GenCapP < LoadMaxP$ .

8.5.6 Sheddable Load

Sheddable load or demand-managed load is a method of controlling low priority loads that may be switched off.

<sup>4</sup> That is, the maximum Load up to that point in the simulation

Asim provides a simple implementation of sheddable loads based on average instantaneous generator load factor:

- If the load factor is greater than a percentage of maximum loading (ShedLoadPct), then the sheddable load is limited as much as possible to bring the load factor back to ShedLoadPct
- If the load factor is less than the percentage of maximum loading, the sheddable load is not limited
- A sheddable load takes a defined amount of time to react (ShedLoadT). All sheddable load calculations are delayed by this time.
- Currently, only one sheddable load can be used. Numerous loads must be aggregated in the current implementation of Asim.

To use the sheddable load function, configure Asim as follows:

1. Configure sheddable load parameter ShedIdealPct (eg. To 99%),
2. Provide a sheddable load profile (ShedLoadP) in [time,value] format along with the normal LoadP and other input data
3. Run **Asim**. The results will show that the sheddable load component is limited according to the rules above.



The sheddable load profile ShedLoadP is assumed to be a portion of LoadP. For example, if the instantaneous load is 100kW, and the sheddable load is 20kW, then the non-sheddable portion of load is 80kW.

The instantaneous load that is being shed is shown in ShedOffP, and the energy required over the simulation that was not provided is summed in ShedE.



It is important to note that sheddable loads affect the spinning reserve constraint. Please read the following section carefully.

With sheddable load, spinning reserve is calculated in one of the following ways:

1. Allow online sheddable load to reduce *StatSpinSetP* and *PvCoverage* down to zero, given that this load can be shed if required (default)
2. Maintain a minimum of *StatSpinSetP* at all times

The *maintain StatSpinSetP method* can be selected by setting *StatMaintainSpin* to any positive value.

In the default method the actual reserve requirement is calculated from the maximum of both PvCoverage and the static StatSpinSetP, offset by any sheddable load requirements

$$reserve = \begin{matrix} 0, \\ MAX \left( MAX(StatSpinSetP, PvCoverage) - ShedP + ShedOffP \right) \end{matrix}$$

In the *maintain StatSpinSetP method*, StatSpinSetP is always maintained, and any sheddable load requirements offset PvCoverage:

$$reserve = \begin{matrix} StatSpinSetP, \\ MAX \left( PvCoverage - ShedP + ShedOffP \right) \end{matrix}$$

The Generator setpoint GenCfgSetP, actual spinning reserve and other calculations are the same as described in *8.5.1 Spinning Reserve*.

Note that LoadP is a “simulated” LoadP, which is actually the input Load as provided in a csv file, minus ShedOffP, so that Asim can switch the sheddable load on and off as required.

The parameters used to configure sheddable loads, and their output variables, are described in further detail in Section 11 “Parameter Reference” on page 26.

### 8.5.7 Generator Setpoint filter

The generator setpoint “GenCfgSetP” (ie. the minimum online capacity required) is calculated to cover a percentage (PvCvgPct) of solar output (PvP) and spinning reserve (StatSpinSetP). GenCfgSetP is then filtered using an Infinite Impulse Response filter. The filter coefficient GenCfgSetK may be customised to tune the response of the filter, as long as:

- $0 \leq GenCfgSetK < 1$

The response characteristics of the filter can be seen in the chart on Page 20.

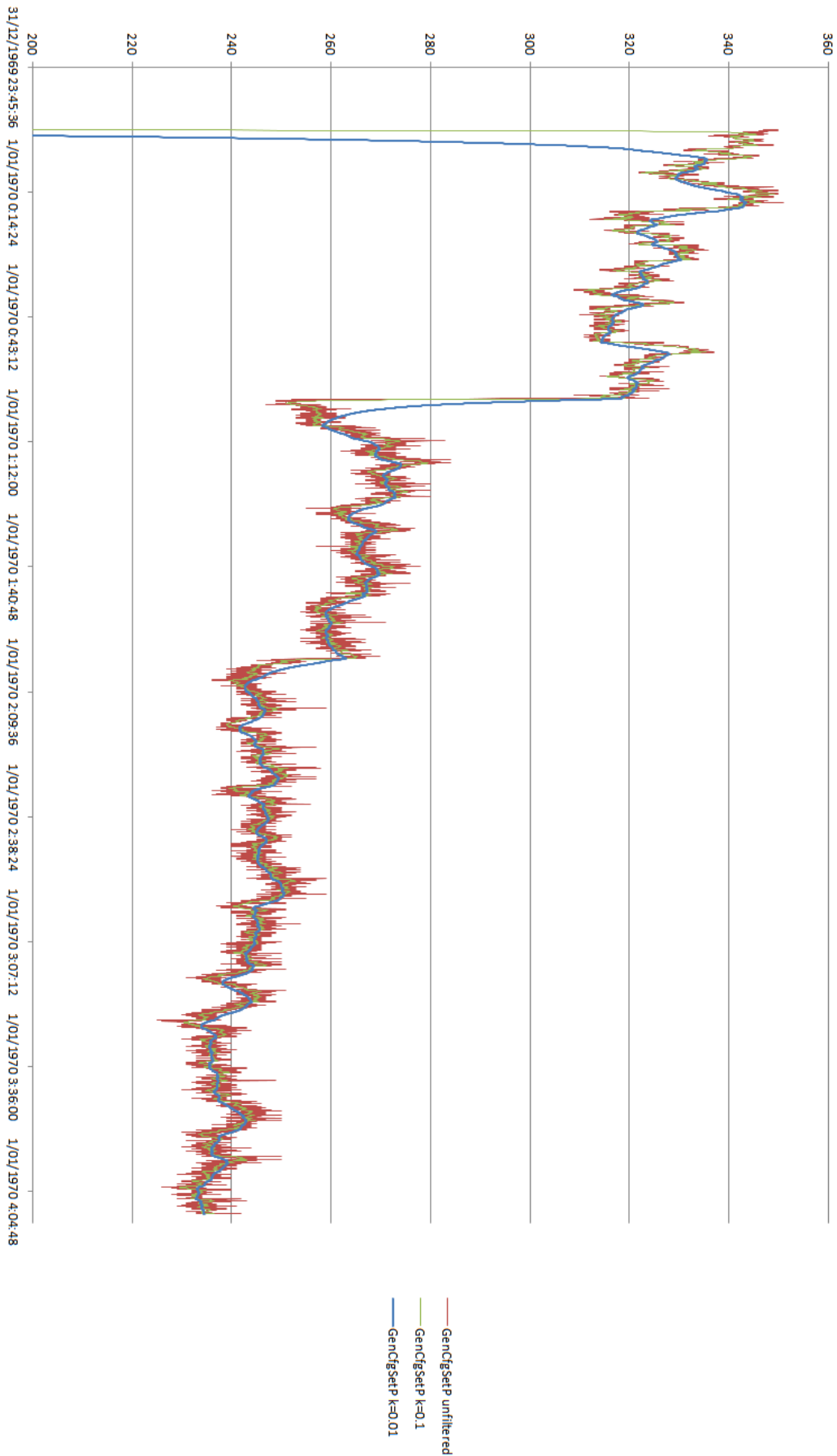


Figure 6 GenCfgSet Filter characteristics

### 8.5.8 Service Intervals

Service intervals can be used to simulate the down-time associated with servicing a generator over a long period of time.

- Up to 6 service intervals can be set per generator
- Each service interval has its own outage time
- Services that occur at the same time are joined into one service for the sum of the outage times
- One service counter counts all services per generator

For Generator 1, the service variables are:

Name	Description
Gen1Service1T	The number of run hours after which generator 1 must be serviced for service 1
Gen1Service1OutT	The number of hours generator 1 will be offline for service 1
Gen1Service#T	The number of run hours after which generator 1 must be serviced for service #, where # is 1-6
Gen1Service#OutT	The number of hours generator 1 will be offline for service #, where # is 1-6
Gen1ServiceCnt	The total number of all services performed on the generator

If two service intervals are defined for generator 1 at 1000 hours and 10,000 hours, with a service time of 2 and 5 hours respectively, then the generator will be taken offline at the following run hours:

Elapsed hours	Gen 1 Run Hours	Offline time (this service)	Gen 1 Service Counter
...			
999	999		0
1000	1000		0
1002	1000	2 hours	1
...			
2001	1999		1
2002	2000		1
2004	2000	2 hours	2
...			
10017	9999		9
10018	10000		9
10025	10000	7 hours	10
...			

Even though there are two services joined together at 10,000 run hours for a total service time of 7 hours, the service counter is only incremented once.

## 8.6 Generator Validation

It is possible to replace the generator simulator with a simple validation mode. This is used to calculate the following variables based on actual kW readings:

- Run Counter, Start Count & Stop Count
- Energy & Fuel Used Counters
- Load Factor, Ideal Loading, Overload status
- Total Generator output, Total spinning reserve

To use this mode, set up **Asim** as follows:

1. Supply input files with real generator load data for individual generators, i.e. Gen1P, Gen2P, etc. in csv format
2. Configure the following parameters to match the site:
  - a. Fuel Efficiency (Gen#FuelCons\*P, Gen#FuelCons\*L)
  - b. generator rating (Gen#MaxP)
3. Choose from the calculations listed above to write to an output file (eg Run Count and Start Count)
4. Supply the parameter “GeneratorStats” – see the section “Program Options” on page 5.
5. Run **Asim** with the desired start time & number of iterations



Do not supply solar data or station load data, as **Asim** will ignore these since the generator outputs are already determined.

You may now compare the counters you selected in step 4. with their corresponding values from a simulation run of the same site.

The Validation mode will use the following conditions to calculate state:

Condition	State
GenP < 5kW	Generator is Off
GenP > 10kW	Generator is On

The following state & transitions will increment the respective counter:

State / Transition	Incremented Counter
Gen On > Off	GenStopCnt
Gen Off > On	GenStartCnt
Gen On	GenRunCnt, GenFuelUsedCnt

## 8.7 Run Time Extensions

Asim can be extended with simple code blocks that are evaluated on every iteration, called Run Time Extensions (RTEs). An understanding of C# is required. To use this RTEs, create a text file containing the expression to evaluate.



- Each file must be written in C# (which is beyond the scope of this document)
- Each file may contain multiple lines of code, however each line must be a single-line expression that is not evaluated in the scope of the other lines of code
- Each code block may reference shared parameters in the simulation, such as LoadP or Gen1P
- Each line is evaluated on each iteration, after input files are read and before any other Asim functions are evaluated
- You may use the special variable '*it*' to refer to the current iteration counter

For example, to create a new variable to track the excess renewable energy that could be used to charge a battery, BatSetP, which is 80% of the PV spill output, do the following:

1. All shared parameters must exist before they are used by RTEs, so create the new parameter by adding it in an input file, called rte.csv:

```
t, BatSetP
0, 0
```

Figure 7 File 'rte.csv'

2. Create another text file called rte.txt containing the expression to set the value of BatSetP:

```
BatSetP = PvSpillP * 0.8D
```

Figure 8 File 'rte.txt'

3. Configure the Analyser to run your simulation, and include both rte.csv and rte.txt:

	A	B	C	D
1	config			
2	FlattenApplication	..\bin\AsimExcelTools.exe		
3	Simulator	..\bin\Asim.exe		
4	directory	.		
5	Start Time	14/10/2012 14:25:00		
6	iterations	86400		
7	input	load.csv		
8	input	rte.csv		
9	output	output.csv	600	BatSetP
10	eval	rte.txt		
11				
12				

In the example, a file called output.csv will be created with 10 minute data containing the value of BatSetP statistics for each 10 minute block. The simulation will be run for one day.

## 9 Importing data from other applications

Data can be imported from any application capable of creating a CSV file in the recognised format (see “File Formats” above). This may require some manual editing. There is no limit to the size or number of rows to the csv file.

## 10 Modifying

### 10.1 License & Copyright

All modifications must adhere to the GNU General Public License. See section 2 *License & Warranty* for more information.

Asim is the copyright of Power And Water Corporation.

### 10.2 Obtaining the Source Code

Source code is hosted in GitHub at <https://github.com/thinkOfaNumber/Asim>. Using GitHub and the Git version control system are beyond the scope of this manual.

### 10.3 Submitting patches and bug fixes

Please submit changes by using a pull request on GitHub.

All pull requests must include:

- Formatted and tested code
- Updated documentation
- Unit tests to thoroughly test the behaviour and edge cases of your feature / bugfix

### 10.4 Editing the source code



It is recommended that Microsoft Visual Studio 2010 or higher is used to modify and compile the source code. Free versions are available for download at Microsoft.com. Specifics of using Visual Studio, C#, and .NET are beyond the scope of this manual.

There are some important parts to modifying **Asim** code:

1. Sharing variables
2. Implementing the IActor interface
3. Order of operations
4. Performance
5. Unit testing



## 10.5 Sharing Variables

Variables are shared using a hash table or dictionary, which is internally managed by the SharedContainer class. This class stores a dictionary of Shared objects, which boxes a double-precision floating point value in a reference type.

SharedContainer is implemented using a Singleton pattern. To get an instance, use the static Instance property. Create variables needed by your module by using the method GetOrNew() passing the name of the variable. GetOrNew() returns a reference to the Shared object in the dictionary, or creates a new entry if it doesn't exist and references that:

```
1    SharedContainer sharedContainer = SharedContainer.Instance;
2    Shared genP = sharedContainer.GetOrNew("GenP");
```

You can now update the Val property of this object in any module independently, and all other references to this object will be updated as well:

```
3    genP.Val = 7;
```

There are other functions available in the SharedContainer and the Shared class for scaling; events; glob matching, etc. It is recommended to follow the “write-once read-many” pattern. In other words, only one object should write to a Shared object, but many objects can read from it.

## 10.6 Implementing the IActor interface

Any new control system tasks must implement IActor. This provides an interface for the following methods:

```
4    public interface IActor
5    {
6        void Run(ulong iteration);
7        void Init();
8        void Finish();
9    }
```

Implement these methods for your task. They are run in the following order:

1. All Init() methods of all actors are executed<sup>5</sup>.
2. For the number of iterations specified, all Run() methods are executed<sup>5</sup>.
3. All Finish() methods of all actors are executed<sup>5</sup>.



All shared variables used by an actor must be set-up in the Init() method.

---

<sup>5</sup>Actors will be run in the order added to the actors list, and not in parallel.

## 10.7 Order of Operations

**Asim** is deterministic in the sense that multiple instances with the same inputs will produce exactly the same output. In order to achieve this, no operations are executed in parallel. The current order of control system tasks is:

1. All variables are updated from the provided input files
2. Any Run Time extensions are evaluated
3. System Load is derived from LoadP, LoadMaxLimP (if set), and ShedOffP (if any)
4. Station specific functions
5. Sheddable load specific functions
6. Generator specific functions
7. Solar specific functions
8. Statistics are calculated and all output variables and statistics are written

## 10.8 Performance & Speed

The performance of **Asim** depends on a few factors:

1. The complexity of the simulated control system
2. The speed of the host system
3. The number of file reads and writes (variables, statistics and output period) per iteration, especially for output files written every iteration
4. The number of watch variable outputs
5. The number of analyser templates
6. The number of Run Time Extensions

The speed of **Asim** grows linearly with the number of iterations.

To improve the speed, reduce the number of analyser templates; reduce the number of watched variables; reduce the number of output files; reduce the number of variables or statistics in an output file; or increase the output file period.

# 11 Parameter Reference

The existing shared parameters are defined here:

Name	Description
<i>inputs:</i>	
LoadP	Load/Demand
LoadMaxLimP	If positive, this is applied as a maximum limit to the Load/Demand (LoadP)
LoadMaxUpP	This maximum rate of positive change (kW/s) is applied to the input load profile (LoadP) before being used by the simulator. This can be used (for example) to smooth 10 minute data with large load steps so-as not to black out the simulator.
LoadMaxDownP	This maximum rate of negative change (kW/s) is applied to the input load profile (LoadP) before being used by the simulator.
StatHystP	Hysteresis
StatSpinSetP	Spinning Reserve Setpoint
StatMaintainSpin	Set to 0 to allow spinning reserve to be offset by sheddable load; Set to 1 to maintain StatSpinSetP at all times
GenBlackCfg	Black Start Configuration
GenAvailCfg	Healthy Generators
GenMinRunTPa	Generator Minimum Run Time
GenSwitchDownDelayT	Generator Switch-down Delay Time. Set this to delay generator switch-downs, until GenCfgSetP < lower configuration power for this amount of time
Gen#MaxP	Generator # Nominal Rating. # represents Gen ID (1-8)
Gen#MinRunTPa	Generator # Minimum Run Time
Gen#IdealPctP	Generator # Ideal Load Setpoint (% of MaxP)
Gen#Service#T	Generator # Service Interval (h). After this amount of run hours have elapsed, the generator is taken offline for Gen#Service#OutT hours.
Gen#Service#OutT	The time (h) it takes to perform a service.
GenConfig#	Configuration table, Row #. # represents Configuration ID (1-256)

PvAvailP	Available Solar Energy
PvMaxLimP	If positive, this is applied as a maximum limit to the available solar energy (PvAvailP)
ShedLoadT	Load shed latency for sheddable load
ShedIdealPct	Ideal load factor to maintain by limiting sheddable loads
ShedLoadP	Size of sheddable load
<i>outputs:</i>	
StatP	Station Output
StatBlackCnt	Number of black starts
StatSpinP	Actual Spinning Reserve
GenP	Total Generator Output
GenMaxP	Total Capacity Online
GenMinRunT	Minimum Run Time remaining
GenOnlineCfg	Generators Online
GenSetCfg	Generators Requested to be Online
GenCfgSetP	Generator Configuration Setpoint
GenOverload	Any Generators Overloaded
GenIdealP	Total Ideal Load (kW) of online generators
GenSpinP	Total Spinning Reserve of online generators
Gen#LoadFact	Generator # Load Factor (between 0-1)
Gen#P	Generator # Actual Power Output
Gen#StartCnt	Generator # Number of Starts
Gen#StopCnt	Generator # Number of Stops
Gen#RunCnt	Generator # Run time (h)
Gen#E	Generator # energy produced (kWh)
Gen#FuelCnt	Generator # Fuel Used (L)
Gen#IdealP	Generator # Ideal Load (kW)

## CHAPTER 11 PARAMETER REFERENCE

Gen#ServiceCnt	The total number of services on this generator.
PvE	Total Solar output (kWh)
PvAvailE	Total Available Solar (kWh)
PvSpillE	Total Spilt Solar (kWh)
PvSetMaxDownP	Setpoint Maximum positive ramp rate (kW/s)
PvSetMaxUpP	Setpoint Maximum negative ramp rate (kW/s)
PvSetP	Actual solar farm setpoint
PvSpillP	Unused solar energy
PvP	Actual solar output
ShedLoadP	Sum of all sheddable loads, regardless of their current state
ShedP	Sum of online sheddable loads
ShedOffP	Sum of offline portion of sheddable loads
ShedE	Sum of accumulated energy that was required to bring offline sheddable load back online