MANUAL

INL/EXT-20-58976, GDE-939
Revision 1
Printed May 12, 2022

HERON User Manual

Cristian Rabiti, Paul W. Talbot, Abhinav Gairola, Jia Zhou

Prepared by
Idaho National Laboratory
Idaho Falls, Idaho 83415

Idaho National Laboratory

INL/EXT-20-58976, GDE-939
Revision 1
Printed May 12, 2022

HERON User Manual

Project Manager:
Cristian Rabiti
Principal Investigator and Technical Leader:
Paul W. Talbot
Main Developers:
Paul W. Talbot
Abhinav Gairola
Additional Developers:
Jia Zhou

Contents

1 Introduction

HERON is a generic software plugin for RAVEN to perform stochastic technoeconomic analysis of grid energy-resource systems with economic drivers. The development targets analysis of electricity and secondary product generation and consumption in regional balancing areas, including flexibility to include arbitrary resources as well as arbitrary resource consumers and producers. HERON is developed to drive optimization via economic drivers such as system cost minimization, profitability, and net present value (NPV) maximization. As a plugin of RAVEN, HERON provides two primary functions: the automatic generation of RAVEN workflows, and models for optimizing high-resolution dispatch of arbitrary systems including resources, resource consumers, and resource producers. HERON leverages the synthetic history training and generation tools, sampling workflows, code Application Programming Interfaces (API), and optimization schemes.

2 Installation and how to run

2.1 Installation

As a plugin of RAVEN, HERON is installed as a submodule. RAVEN maintains up-to-date instructions for plugin installation in its documentation (see the link to the Raven plugin installation.

2.2 How to run

Directly running HERON through RAVEN has not finished implementation. To run HERON, use bash to run the executable:

```
/path/to/HERON/heron my_heron_input.xml
```

replacing

- `/path/to/HERON` with your path to the HERON repository; and
- `my_heron_input.xml` with your HERON XML input.

For example, if my HERON is located as a submodule of RAVEN in `/projects/raven`, and I wanted to run the integration test `production_flex`, I could do the following:

```
cd ~/projects/raven/plugins/HERON/tests/integration_tests/production_flex
~/projects/raven/plugins/HERON/heron heron_input.xml
```

Alternatively, you can use Python to run `HERON/src/main.py` with the HERON XML input as argument; however, this will bypass loading the `raven_libraries` and other initialization.

3 Input Structure

In the following sections we describe the input structure in a general sense, with details in following chapters.

3.1 XML Input

HERON makes use of the eXtensible Markup Language (XML) for its input structure, similar to RAVEN. XML is made up of nodes, which have parameters and subnodes. For example:

```
<node_tag par_name="par_value">
  <sub_tag sub_par_name="sub_par_value">sub_value</sub_tag>
</node_tag>
```

The node's name (or tag) opens the XML element. In the example, we have two nodes, named **<node_tag>** and **<sub_tag>**. The node **<node_tag>** has a parameter with name **par_name**. The parameter **par_name** has the value **'par_value'**. Similarly, the **<sub_tag>** node has a parameter and corresponding value. The **<sub_tag>** further has a value itself given by **'sub_value'**.

We will use the terminology **<node>**, **<subnode>**, **parameter**, and **'value'** to describe the input structure of HERON.

3.2 Structure

The HERON XML Input makes use of three main nodes within the root node **<HERON>**:

- **<Case>**, in which general features of the desired solve are described, including general economics to apply, simulation properties such as project length and time stepping, and so forth.
- **<Components>**, in which the components of the grid system to analyze are defined, including their physical processes and economics.
- **<DataGenerators>**, in which data manipulation tools such as synthetic history generators and custom code functions.
- **<TestInfo>**, which is reserved for regression tests in HERON, describes the purpose of the test and information about the test's implementation.

Details for the three nodes used in HERON analyses are enumerated the following sections.

4 Cases

Each HERON analysis is referred to as a “case”. The **<Case>** node is used to describe the general elements of the analyis, such as the firm economics, dispatch time stepping, and so forth.

HERON relies on this **<xml>** node which informs the algorithm as to how the case has to be processed by using the predefined metrics described in the following sections.

4.1 Case

The **<Case>** node contains the general physics and economics information required for a HERON workflow to be created and solved.

The **<Case>** node recognizes the following parameters:

- `name`: ***string, required***, the name by which this analysis should be referred within HERON.

The **<Case>** node recognizes the following subnodes:

- **<label>**: provides static label information to the model; unused in computation. These data will be passed along through the meta class and output in the simulation result files. These data can also be accessed within user-defined transfer functions by using `meta['HERON']['Case`
  The **<label>** node recognizes the following parameters:
  - `name`: ***string, optional***, the generalized name of the identifier. Example: “¡label name=”state”¿Idaho¡/label¿”
- **<mode>**: ***[opt, sweep]***, determines the mode of operation for the outer/inner RAVEN. If “sweep” then parametrically sweep over distributed values. If “opt” then search distributed values for economic metric optima.
- **<verbosity>**: ***[silent, quiet, all, debug]***, determines the level of verbosity for the outer/inner RAVEN runs.
  Default: all. If “silent” only errors are displayed. If “quiet” errors and warnings are displayed. If “all” errors, warnings, and messages are displayed. If “debug” errors, warnings, messages, and debug messages are displayed.
- **<debug>**: Including this node enables a reduced-size run with increased outputs for checking how the sampling, dispatching, and cashflow mechanics are working for a particular input. Various options for modifying how the debug mode operates are included for convenience; however, just including this node will result in a minimal run.

  The **<debug>** node recognizes the following subnodes:

    - `<inner_samples>`: ***integer***, sets the number of inner realizations of the stochastic synthetic histories and dispatch optimization to run per outer sample. Overrides the `<num_arma_steps>` option while `<debug>` mode is enabled.
      Default: 1
    - `<macro_steps>`: ***integer***, sets the number of macro steps (e.g. years) the stochastic synthetic histories and dispatch optimization should include.
      Default: 1
    - `<dispatch_plot>`: ***[True, Yes, 1, False, No, 0, t, y, 1, f, n, 0]***, provides a dispatch plot after running through `<inner_samples>` and `<macro_steps>` provided. To prevent plotting output during debug mode set to "False".
      Default: True

- `<parallel>`: Describes how to parallelize this run.

  The `<parallel>` node recognizes the following subnodes:

    - `<outer>`: ***integer***, the number of parallel runs to use for the outer optimization run. The product of this number and `<inner>` should be at most the number of parallel process available on your computing device. This should also be at most the number of samples needed per outer iteration; for example, with 3 opt bound variables and using finite differencing, at most 4 parallel outer runs can be used.
      Default: 1
    - `<inner>`: ***integer***, the number of parallel runs to use per inner sampling run. This should be at most the number of denoising samples, and at most the number of parallel processes available on your computing device.
      Default: 1

- `<data_handling>`: Provides options for data handling within HERON operations.

  The `<data_handling>` node recognizes the following subnodes:

    - `<inner_to_outer>`: ***[csv, netcdf]***, which type of data format to transfer results from inner (stochastic dispatch optimization) runs to the outer (capacity and meta-variable optimization) run. CSV is generally slower and not recommended, but may be useful for debugging. NetCDF is more generally more efficient.
      Default: netcdf

- `<num_arma_samples>`: ***integer***, provides the number of synthetic histories that should be considered per system configuration in order to obtain a reasonable representation of the economic metric. Sometimes referred to as "inner samples" or "denoisings".

- `<time_discretization>`: node that defines how within-cycle time discretization should be handled for solving the dispatch.

  The `<time_discretization>` node recognizes the following subnodes:

    - `<time_variable>`: ***string***, name for the `time` variable used in this simulation.
      Default: time
    - `<year_variable>`: ***string***, name for the `year` or `macro` variable used in this simulation.
      Default: Year
    - `<start_time>`: ***float***, value for `time` variable at which the inner dispatch should begin.
      Default: 0
    - `<end_time>`: ***float***, value for `time` variable at which the inner dispatch should end. If not specified, both `<time_interval>` and `<num_timesteps>` must be defined.
    - `<num_steps>`: ***integer***, number of discrete time steps for the inner dispatch. Either this node or `<time_interval>` must be defined.
    - `<time_interval>`: ***float***, length of a time step for the inner dispatch, in units of the time variable (not indices). Either this node or `<num_timesteps>` must be defined. Note that if an integer number of intervals do not fit between `<start_time>` and `<end_time>`, an error will be raised.

- `<economics>`: node containing general economic setting in which to perform HERON analysis.

  The `<economics>` node recognizes the following subnodes:

    - `<ProjectTime>`: ***float***, the number of cycles (usually years) for the HERON analysis to cover.
    - `<DiscountRate>`: ***float***, rate representing the time value of money to the firm used to discount cash flows in the multicycle economic analysis. Passed to the CashFlow module.
    - `<tax>`: ***float***, the taxation rate, a metric which represents the rate at which the firm is taxed. Passed to the CashFlow module.
    - `<inflation>`: ***float***, a metric which represents the rate at which the average price of goods and services in an economy increases over a cycle, usually a year. Passed to the CashFlow module.
    - `<verbosity>`: ***integer***, the level of output to print from the CashFlow calculations. Passed to the CashFlow module.

- `<dispatcher>`: This node defines the dispatch strategy and options to use in the "inner" run.

  The `<dispatcher>` node recognizes the following subnodes:

    - `<pyomo>`: The `pyomo` dispatcher uses analytic modeling and rolling windows to solve dispatch optimization with perfect information via the pyomo optimization library.

The `<pyomo>` node recognizes the following subnodes:

- ◦ `<rolling_window_length>`: ***integer***, Sets the length of the rolling window that the Pyomo optimization algorithm uses to break down histories. Longer window lengths will minimize boundary effects, such as nonoptimal storage dispatch, at the cost of slower optimization solves. Note that if the rolling window results in a window of length 1 (such as at the end of a history), this can cause problems for pyomo.
  Default: 24
- ◦ `<debug_mode>`: ***[True, Yes, 1, False, No, 0, t, y, 1, f, n, 0]***, Enables additional printing in the pyomo dispatcher. Highly discouraged for production runs.
  Default: False.
- ◦ `<solver>`: ***string***, Indicates which solver should be used by pyomo. Options depend on individual installation.
  Default: 'glpk' for Windows, 'cbc' otherwise.

◦ `<custom>`: – no description yet –

The `<custom>` node recognizes the following subnodes:

- ◦ `<location>`: ***string***, The hard drive location of the custom dispatcher. Relative paths are taken with respect to the HERON run location. Custom dispatchers must implement a `dispatch` method that accepts the HERON case, components, and sources; this method must return the activity for each resource of each component.

◦ `<validator>`: This node defines the dispatch validation strategy and options to use in the "inner" run.

The `<validator>` node recognizes the following subnodes:

◦ `<Example>`: Uses a demonstration-only validator that constrains the change for any resource to a constant "delta".

The `<Example>` node recognizes the following subnodes:

- ◦ `<delta>`: ***float***, the maximum absolute change in any resource between successive time steps.
- ◦ `<tolerance>`: ***float***, the strictness with which the constraint should be enforced. Note that some small numerical exception is expected.

◦ `<optimization_settings>`: node that defines the settings to be used for the optimizer in the "outer" run.

The `<optimization_settings>` node recognizes the following subnodes:

◦ `<metric>`: ***[expectedValue, minimum, maximum, median, variance, sigma, percentile, variationCoefficient, skewness, kurtosis, sharpeRatio, sortinoRatio, gainLossRatio, expectedShortfall, valueAtRisk]***, determines the statistical metric (calculated by RAVEN BasicStatistics or EconomicRatio PostProcessors) from the "inner" run to be used as the objective in the "outer" optimization.

        - For "percentile" the additional parameter *percent='X'* is required where *X* is the requested percentile (a floating point value between 0.0 and 100.0).
        - For "sortinoRatio" and "gainLossRatio" the additional parameter *threshold='X'* is required where *X* is the requested threshold ('median' or 'zero').
        - For "expectedShortfall" and "valueAtRisk" the additional parameter *threshold='X'* is required where *X* is the requested α value (a floating point value between 0.0 and 1.0).

      The **`<metric>`** node recognizes the following parameters:

        - **`percent`**: ***float, optional***, requested percentile (a floating point value between 0.0 and 100.0). Required when **`<metric>`** is "percentile."
          Default: 5
        - **`threshold`**: ***string, optional***,
            - requested threshold ('median' or 'zero'). Required when **`<metric>`** is "sortinoRatio" or "gainLossRatio."
              Default: 'zero'
            - requested α value (a floating point value between 0.0 and 1.0). Required when **`<metric>`** is "expectedShortfall" or "valueAtRisk."
              Default: 5.0

    - **`<type>`**: ***[min, max]***, determines whether the objective should be minimized or maximized.
        - when metric is "expectedValue," "minimum," "maximum," "median," "percentile," "sharpeRatio," "sortinoRatio," "gainLossRatio"
          Default: max
        - when metric is "variance," "sigma," "variationCoefficient," "skewness," "kurtosis," "expectedShortfall," "valueAtRisk"
          Default: min

- **`<dispatch_vars>`**: This node defines a set containing additional variablesto sample that are not associated with a specific component.

  The **`<dispatch_vars>`** node recognizes the following subnodes:

    - **`<variable>`**: This node defines the single additional dispatch variable used in the case.

      This value can be taken from any *one* of the sources as subnodes (described below): **`<fixed_value>`**, **`<sweep_values>`**, **`<opt_bounds>`**, **`<variable>`**, **`<ARMA>`**, **`<Function>`**, **`<ROM>`**. The **`<variable>`** node recognizes the following parameters:

        - **`name`**: ***string, optional***, The unique name of the dispatch variable.

      The **`<variable>`** node recognizes the following subnodes:

- **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
- **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
- **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
- **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
- **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
  - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
- **<ROM>**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **<ROM>** node recognizes the following parameters:
  - **rom**: ***string, required***, indicates which DataGenerator ROM should be used for this value.
  - **variable**: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

  The **<ROM>** node recognizes the following subnodes:
  - **<input>**: designates the source of one of the inputs to the ROM.

    This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**. The **<input>** node recognizes the following parameters:
    - **name**: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

    The **<input>** node recognizes the following subnodes:
    - **<fixed_value>**: ***float***, indicates this value should be fixed in the outer

run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:

    - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.

  - **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:

    - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.

  - **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:

    - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.

  - **<variable>**: ***string***, the name of the variable from inner RAVEN variables.

  - **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:

    - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.

  - **<Function>**: ***string***, indicates this value should be taken from a Python function, as described in the **<DataGenerators>** node. The **<Function>** node recognizes the following parameters:

    - **method**: ***string, optional***, the name of the **<DataGenerator>** from which this value should be taken.

  - **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
    Default: 1

- **<Function>**: ***string***, indicates this value should be taken from a Python function, as described in the **<DataGenerators>** node. The **<Function>** node recognizes the following parameters:

  - **method**: ***string, optional***, the name of the **<DataGenerator>** from which this value should be taken.

- **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
  Default: 1

5 Components

The **<Components>** node contains the technical and economic definitions of the components (or units) within the grid system that needs to be solved. The **<Components>** node contains many **<Component>** nodes defining each of the grid components.

5.1 Component Activity

At each time point in a dispatch optimization, HERON attempts to calculate the most optimal usage of each component in the system. To report the usage of each resource in each unit in the system, HERON uses some conventions that we explain here.

We can consider two quantities when reporting the activity of a component over time: the quantity of product consumed or produced, or the rate at which a product is consumed or produced. For instance, if a Component produces "widgets", at a particular hour we can either talk about the total number of widgets produced during an hour period, or about the rate (in widgets per second) at which they were produced. Generally, we only track the rate of production for all Components when performing dispatch optimization; however, for a **<Component>** who **<stores>** a resource, we track the *level* (or *net quantity produced*) instead of the *rate*.

5.1.1 General Activity

For most components, when the activity is reported at a particular time for a component, it represents the instantaneous production level of that component at that time. For example, if a component's max capacity is given as 10 widgets per second, and the activity reported for hour 2 is 3, then instantaneously at the beginning of hour 2 the unit is assumed to be producing 3 widgets per second. Simlarly, if a power plant has a capacity of 100 MW and activity is reported as 5 for a particular hour, then at beginning of that hour the plant is producing 5 MW. Between time points, it is assumed that the rate of production varies linearly between the reported time points.

5.1.2 Storage Activity

Unlike other components, for a storage (a **<Component>** who **<stores>** a resource) we report the instantaneous *level* of the storage instead of the instantaneous production rate. We formally define the level (or "net quantity produced") at a specific time as

$$L(t) = L_0 + \int_{t_0}^{t} R(t)dt, \qquad (1)$$

where:

- L is the net quantity produced at time t,
- L_0 is an initial level at the start of the analysis period,
- t is the continuous variable for *time*,
- t_0 is the starting time, at which the components has level L_0, and
- $R(t)$ is the rate of production for the component at time t.

In discrete terms,

$$L_k = L_0 + \sum_{i=1}^{k-1} R_i \Delta_k, \tag{2}$$

where:

- k is an index for discrete time intervals t_k,
- L_k is the level at the beginning of time interval t_k,
- R_k is the production rate for the component during time interval t_k, and
- Δ_k is the length of the time interval t_k.

The equivalent continuous production rate for a storage component is therefore

$$R(t) = \frac{\partial}{\partial t} L(t), \tag{3}$$

or, in discrete form given the constant production assumption,

$$R_k = \frac{L_{k+1} - L_k}{\Delta_k}. \tag{4}$$

We want to emphasize this key difference: the *production rate* is interpreted differently for storage and other producers. For **storages**,

- production rate during a time step is considered *discontinuously constant*, and
- activity is reported as the *level* of the storage at the beginning of the time step.

For **all other components**,

- production rate during a time step is considered *continuously linear*, and
- activity is reported as the *instantaneous production rate* of the storage at the beginning of the time step.

This difference has significant impact when developing a **<Validator>** or plotting dispatch results. As a demonstration, see Figures 1 and 2.

In Figure 1, the black dots indicate the statepoints that HERON reports, while the green dashed line is the assumed production between statepoints. Note the y-axis is in units of production rate.

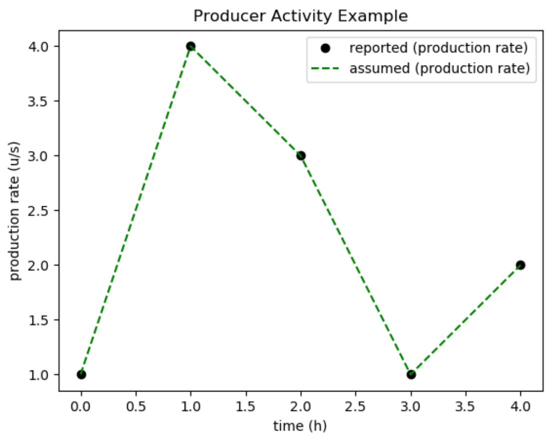

Figure 1: Producer Example

In Figure 2, the black dots indicate the statepoints that HERON reports, with units of emphlevel, not production rate. The red dotted line is the assumed level between statepoints, and the green dashed lines are the equivalent production between each statepoint. Note the dual y-axis: one is in units of production rate (green, left), while the other is in units of level or quantity (red, right). That is, the storage unit filly slowly, then fills quickly, then totally depletes before filling again. This same activity can be represented by the production rates in the dashed green lines, with negative production indicating absorbing into storage and positive production indicating emission from storage.

5.2 Component

defines a component as an element of the grid system. Components are defined by the action they perform such as **<produces>** or **<consumes>**; see details below.

The **<Component>** node recognizes the following parameters:

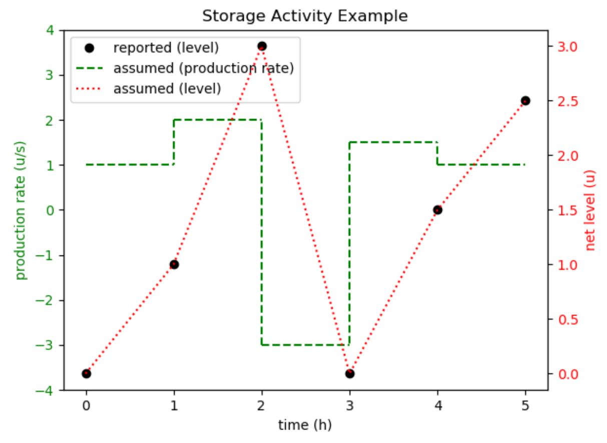

Figure 2: Storage Example

- `name`: ***string, required***, identifier for the component. This identifier will be used to generate variables and relate signals to this component throughout the HERON analysis.

The `<Component>` node recognizes the following subnodes:

- `<produces>`: indicates that this component produces one or more resources by consuming other resources. The `<produces>` node recognizes the following parameters:
  - `resource`: ***comma-separated strings, required***, the resource produced by this component's activity.
  - `dispatch`: ***[fixed, independent, dependent], optional***, describes the way this component should be dispatched, or its flexibility. `fixed` indicates the component always fully dispatched at its maximum level. `independent` indicates the component is fully dispatchable by the dispatch optimization algorithm. `dependent` indicates that while this component is not directly controllable by the dispatch algorithm, it can however be flexibly dispatched in response to other units changing dispatch level. For example, when attempting to increase profitability, the `fixed` components are not adjustable, but the `independent` components can be adjusted to attempt to improve the economic metric. In response to the `independent` component adjustment, the `dependent` components may respond to balance the resource usage from the changing behavior of other components.

  The `<produces>` node recognizes the following subnodes:
  - `<capacity>`: the maximum value at which this component can act, in units corresponding to the indicated resource.

This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**, **<ROM>**. The **<capacity>** node recognizes the following parameters:

- resource: ***string, optional***, indicates the resource that defines the capacity of this component's operation. For example, if a component consumes steam and electricity to produce hydrogen, the capacity of the component can be defined by the maximum steam consumable, maximum electricity consumable, or maximum hydrogen producable. Any choice should be nominally equivalent, but determines the units of the value of this node.

The **<capacity>** node recognizes the following subnodes:

- **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
  - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
  - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
  - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
- **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
  - variable: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
- **<ROM>**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **<ROM>** node recognizes the following parameters:
  - rom: ***string, required***, indicates which DataGenerator ROM should be used for this value.
  - variable: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

The <ROM> node recognizes the following subnodes:

- <input>: designates the source of one of the inputs to the ROM.

  This value can be taken from any *one* of the sources as subnodes (described below): <fixed_value>, <sweep_values>, <opt_bounds>, <variable>, <ARMA>, <Function>. The <input> node recognizes the following parameters:

  - name: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

  The <input> node recognizes the following subnodes:

  - <fixed_value>: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The <fixed_value> node recognizes the following parameters:
    - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
  - <sweep_values>: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The <sweep_values> node recognizes the following parameters:
    - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
  - <opt_bounds>: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The <opt_bounds> node recognizes the following parameters:
    - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
  - <variable>: ***string***, the name of the variable from inner RAVEN variables.
  - <ARMA>: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the <DataGenerator> node. The <ARMA> node recognizes the following parameters:
    - variable: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
  - <Function>: ***string***, indicates this value should be taken from a Python function, as described in the <DataGenerators> node. The <Function> node recognizes the following parameters:
    - method: ***string, optional***, the name of the <DataGenerator> from which this value should be taken.

    - **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1
  - **<Function>**: ***string***, indicates this value should be taken from a Python function, as described in the **<DataGenerators>** node. The **<Function>** node recognizes the following parameters:
    - **method**: ***string, optional***, the name of the **<DataGenerator>** from which this value should be taken.
  - **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
    Default: 1
- **<minimum>**: provides the minimum value at which this component can act, in units of the indicated resource.

  This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**, **<ROM>**. The **<minimum>** node recognizes the following parameters:
  - **resource**: ***string, optional***, indicates the resource that defines the minimum activity level for this component, as with the component's capacity.

  The **<minimum>** node recognizes the following subnodes:
  - **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
    - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
  - **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
    - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
  - **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
    - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
  - **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
  - **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:

  - `variable`: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.

- **`<ROM>`**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **`<ROM>`** node recognizes the following parameters:
  - `rom`: ***string, required***, indicates which DataGenerator ROM should be used for this value.
  - `variable`: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

  The **`<ROM>`** node recognizes the following subnodes:
  - **`<input>`**: designates the source of one of the inputs to the ROM.

    This value can be taken from any *one* of the sources as subnodes (described below): **`<fixed_value>`**, **`<sweep_values>`**, **`<opt_bounds>`**, **`<variable>`**, **`<ARMA>`**, **`<Function>`**. The **`<input>`** node recognizes the following parameters:
    - `name`: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

    The **`<input>`** node recognizes the following subnodes:
    - **`<fixed_value>`**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **`<fixed_value>`** node recognizes the following parameters:
      - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - **`<sweep_values>`**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **`<sweep_values>`** node recognizes the following parameters:
      - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - **`<opt_bounds>`**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **`<opt_bounds>`** node recognizes the following parameters:
      - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - **`<variable>`**: ***string***, the name of the variable from inner RAVEN variables.
    - **`<ARMA>`**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **`<DataGenerator>`** node. The **`<ARMA>`** node recognizes the following parameters:

            - `variable`: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
        - `<Function>`: ***string***, indicates this value should be taken from a Python function, as described in the `<DataGenerators>` node. The `<Function>` node recognizes the following parameters:
            - `method`: ***string, optional***, the name of the `<DataGenerator>` from which this value should be taken.
        - `<multiplier>`: ***float***, Multiplies any value obtained by this parameter by the given value.
          Default: 1
    - `<Function>`: ***string***, indicates this value should be taken from a Python function, as described in the `<DataGenerators>` node. The `<Function>` node recognizes the following parameters:
        - `method`: ***string, optional***, the name of the `<DataGenerator>` from which this value should be taken.
    - `<multiplier>`: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1
- `<consumes>`: ***comma-separated strings***, The producer can either produce or consume a resource. If the producer is a consumer it must be accompnied with a transfer function to convert one source of energy to another.
- `<transfer>`: describes how input resources yield output resources for this component's transfer function.

  This value can be taken from any *one* of the sources as subnodes (described below): `<linear>`, `<Function>`.

  The `<transfer>` node recognizes the following subnodes:
    - `<Function>`: ***string***, indicates this value should be taken from a Python function, as described in the `<DataGenerators>` node. The `<Function>` node recognizes the following parameters:
        - `method`: ***string, optional***, the name of the `<DataGenerator>` from which this value should be taken.
    - `<linear>`: ***string***, indicates this value should be interpreted as a ratio based on an input value.
      The `<linear>` node recognizes the following subnodes:
        - `<rate>`: ***float***, linear coefficient for the indicated `resource`. The `<rate>` node recognizes the following parameters:
            - `resource`: ***string, optional***, indicates the resource for which the linear transfer rate is being provided in this node.

    - **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1

- **<stores>**: indicates that this component stores one resource, potentially absorbing or providing that resource. The **<stores>** node recognizes the following parameters:

  - **resource**: ***comma-separated strings, required***, the resource stored by this component.
  - **dispatch**: ***[fixed, independent, dependent], optional***, describes the way this component should be dispatched, or its flexibility. `fixed` indicates the component always fully dispatched at its maximum level. `independent` indicates the component is fully dispatchable by the dispatch optimization algorithm. `dependent` indicates that while this component is not directly controllable by the dispatch algorithm, it can however be flexibly dispatched in response to other units changing dispatch level. For example, when attempting to increase profitability, the `fixed` components are not adjustable, but the `independent` components can be adjusted to attempt to improve the economic metric. In response to the `independent` component adjustment, the `dependent` components may respond to balance the resource usage from the changing behavior of other components.

  The **<stores>** node recognizes the following subnodes:

  - **<capacity>**: the maximum value at which this component can act, in units corresponding to the indicated resource.

    This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**, **<ROM>**. The **<capacity>** node recognizes the following parameters:

    - **resource**: ***string, optional***, indicates the resource that defines the capacity of this component's operation. For example, if a component consumes steam and electricity to produce hydrogen, the capacity of the component can be defined by the maximum steam consumable, maximum electricity consumable, or maximum hydrogen producable. Any choice should be nominally equivalent, but determines the units of the value of this node.

    The **<capacity>** node recognizes the following subnodes:

    - **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
      - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.

- **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
- **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
- **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
- **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
  - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
- **<ROM>**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **<ROM>** node recognizes the following parameters:
  - **rom**: ***string, required***, indicates which DataGenerator ROM should be used for this value.
  - **variable**: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

  The **<ROM>** node recognizes the following subnodes:
  - **<input>**: designates the source of one of the inputs to the ROM.

    This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**. The **<input>** node recognizes the following parameters:
    - **name**: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

    The **<input>** node recognizes the following subnodes:
    - **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
      - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the

inner workflow. The **`<sweep_values>`** node recognizes the following parameters:

    - **`debug_value`**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.

  - **`<opt_bounds>`**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **`<opt_bounds>`** node recognizes the following parameters:

    - **`debug_value`**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.

  - **`<variable>`**: ***string***, the name of the variable from inner RAVEN variables.

  - **`<ARMA>`**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **`<DataGenerator>`** node. The **`<ARMA>`** node recognizes the following parameters:

    - **`variable`**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.

  - **`<Function>`**: ***string***, indicates this value should be taken from a Python function, as described in the **`<DataGenerators>`** node. The **`<Function>`** node recognizes the following parameters:

    - **`method`**: ***string, optional***, the name of the **`<DataGenerator>`** from which this value should be taken.

  - **`<multiplier>`**: ***float***, Multiplies any value obtained by this parameter by the given value.
    Default: 1

- **`<Function>`**: ***string***, indicates this value should be taken from a Python function, as described in the **`<DataGenerators>`** node. The **`<Function>`** node recognizes the following parameters:

  - **`method`**: ***string, optional***, the name of the **`<DataGenerator>`** from which this value should be taken.

- **`<multiplier>`**: ***float***, Multiplies any value obtained by this parameter by the given value.
  Default: 1

- **`<minimum>`**: provides the minimum value at which this component can act, in units of the indicated resource.

  This value can be taken from any *one* of the sources as subnodes (described below): **`<fixed_value>`**, **`<sweep_values>`**, **`<opt_bounds>`**, **`<variable>`**, **`<ARMA>`**, **`<Function>`**, **`<ROM>`**. The **`<minimum>`** node recognizes the following parameters:

- **resource**: ***string, optional***, indicates the resource that defines the minimum activity level for this component, as with the component's capacity.

The **<minimum>** node recognizes the following subnodes:

- **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
- **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
  - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
- **<ROM>**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **<ROM>** node recognizes the following parameters:
  - **rom**: ***string, required***, indicates which DataGenerator ROM should be used for this value.
  - **variable**: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

  The **<ROM>** node recognizes the following subnodes:
  - **<input>**: designates the source of one of the inputs to the ROM.

    This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**. The **<input>** node recognizes the following parameters:

    - `name`: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

    The **`<input>`** node recognizes the following subnodes:

    - **`<fixed_value>`**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **`<fixed_value>`** node recognizes the following parameters:
      - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - **`<sweep_values>`**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **`<sweep_values>`** node recognizes the following parameters:
      - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - **`<opt_bounds>`**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **`<opt_bounds>`** node recognizes the following parameters:
      - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - **`<variable>`**: ***string***, the name of the variable from inner RAVEN variables.
    - **`<ARMA>`**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **`<DataGenerator>`** node. The **`<ARMA>`** node recognizes the following parameters:
      - `variable`: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
    - **`<Function>`**: ***string***, indicates this value should be taken from a Python function, as described in the **`<DataGenerators>`** node. The **`<Function>`** node recognizes the following parameters:
      - `method`: ***string, optional***, the name of the **`<DataGenerator>`** from which this value should be taken.
    - **`<multiplier>`**: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1

- **`<Function>`**: ***string***, indicates this value should be taken from a Python function, as described in the **`<DataGenerators>`** node. The **`<Function>`** node recognizes the following parameters:
  - `method`: ***string, optional***, the name of the **`<DataGenerator>`** from which this value should be taken.

    - **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1

- **<initial_stored>**: initial quantity of resource assumed to be present in the storage unit at the beginning of a given calculation, in units of quantity (not rate).
  Default: 0.

  This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**, **<ROM>**.

  The **<initial_stored>** node recognizes the following subnodes:

    - **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
        - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
    - **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
        - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
    - **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
        - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
    - **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
    - **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
        - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
    - **<ROM>**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **<ROM>** node recognizes the following parameters:
        - **rom**: ***string, required***, indicates which DataGenerator ROM should be used for this value.
        - **variable**: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

The **<ROM>** node recognizes the following subnodes:

- **<input>**: designates the source of one of the inputs to the ROM.

  This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**. The **<input>** node recognizes the following parameters:

  - **name**: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

  The **<input>** node recognizes the following subnodes:

  - **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
    - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
  - **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
    - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
  - **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
    - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
  - **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
  - **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
    - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
  - **<Function>**: ***string***, indicates this value should be taken from a Python function, as described in the **<DataGenerators>** node. The **<Function>** node recognizes the following parameters:
    - **method**: ***string, optional***, the name of the **<DataGenerator>** from which this value should be taken.

            - `<multiplier>`: ***float***, Multiplies any value obtained by this parameter by the given value.
              Default: 1
        - `<Function>`: ***string***, indicates this value should be taken from a Python function, as described in the `<DataGenerators>` node. The `<Function>` node recognizes the following parameters:
            - `method`: ***string, optional***, the name of the `<DataGenerator>` from which this value should be taken.
        - `<multiplier>`: ***float***, Multiplies any value obtained by this parameter by the given value.
          Default: 1
    - `<strategy>`: control strategy for operating the storage. If not specified, uses a perfect foresight strategy.

      This value can be taken from any *one* of the sources as subnodes (described below): `<Function>`.

      The `<strategy>` node recognizes the following subnodes:
        - `<Function>`: ***string***, indicates this value should be taken from a Python function, as described in the `<DataGenerators>` node. The `<Function>` node recognizes the following parameters:
            - `method`: ***string, optional***, the name of the `<DataGenerator>` from which this value should be taken.
        - `<multiplier>`: ***float***, Multiplies any value obtained by this parameter by the given value.
          Default: 1
    - `<RTE>`: ***float***, round-trip efficiency for this component as a scalar multiplier.
      Default: 1.0
- `<demands>`: indicates that this component exclusively consumes a resource. The `<demands>` node recognizes the following parameters:
    - `resource`: ***comma-separated strings, required***, the resource consumed by this component.
    - `dispatch`: ***[fixed, independent, dependent], optional***, describes the way this component should be dispatched, or its flexibility. `fixed` indicates the component always fully dispatched at its maximum level. `independent` indicates the component is fully dispatchable by the dispatch optimization algorithm. `dependent` indicates that while this component is not directly controllable by the dispatch algorithm, it can however be flexibly dispatched in response to other units changing dispatch level. For example, when attempting to increase profitability, the `fixed` components are not adjustable, but the `independent` components can be adjusted to attempt to improve

the economic metric. In response to the `independent` component adjustment, the `dependent` components may respond to balance the resource usage from the changing behavior of other components.

The **`<demands>`** node recognizes the following subnodes:

- **`<capacity>`**: the maximum value at which this component can act, in units corresponding to the indicated resource.

  This value can be taken from any *one* of the sources as subnodes (described below): **`<fixed_value>`**, **`<sweep_values>`**, **`<opt_bounds>`**, **`<variable>`**, **`<ARMA>`**, **`<Function>`**, **`<ROM>`**. The **`<capacity>`** node recognizes the following parameters:

  - **`resource`**: ***string, optional***, indicates the resource that defines the capacity of this component's operation. For example, if a component consumes steam and electricity to produce hydrogen, the capacity of the component can be defined by the maximum steam consumable, maximum electricity consumable, or maximum hydrogen producable. Any choice should be nominally equivalent, but determines the units of the value of this node.

  The **`<capacity>`** node recognizes the following subnodes:

  - **`<fixed_value>`**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **`<fixed_value>`** node recognizes the following parameters:
    - **`debug_value`**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
  - **`<sweep_values>`**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **`<sweep_values>`** node recognizes the following parameters:
    - **`debug_value`**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
  - **`<opt_bounds>`**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **`<opt_bounds>`** node recognizes the following parameters:
    - **`debug_value`**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
  - **`<variable>`**: ***string***, the name of the variable from inner RAVEN variables.
  - **`<ARMA>`**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **`<DataGenerator>`** node. The **`<ARMA>`** node recognizes the following parameters:

    - `variable`: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.

- **`<ROM>`**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **`<ROM>`** node recognizes the following parameters:
    - `rom`: ***string, required***, indicates which DataGenerator ROM should be used for this value.
    - `variable`: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

  The **`<ROM>`** node recognizes the following subnodes:
    - **`<input>`**: designates the source of one of the inputs to the ROM.

      This value can be taken from any *one* of the sources as subnodes (described below): **`<fixed_value>`**, **`<sweep_values>`**, **`<opt_bounds>`**, **`<variable>`**, **`<ARMA>`**, **`<Function>`**. The **`<input>`** node recognizes the following parameters:
        - `name`: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

      The **`<input>`** node recognizes the following subnodes:
        - **`<fixed_value>`**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **`<fixed_value>`** node recognizes the following parameters:
            - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
        - **`<sweep_values>`**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **`<sweep_values>`** node recognizes the following parameters:
            - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
        - **`<opt_bounds>`**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **`<opt_bounds>`** node recognizes the following parameters:
            - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
        - **`<variable>`**: ***string***, the name of the variable from inner RAVEN variables.
        - **`<ARMA>`**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **`<DataGenerator>`** node. The **`<ARMA>`** node recognizes the following parameters:

        - `variable`: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
      - `<Function>`: ***string***, indicates this value should be taken from a Python function, as described in the `<DataGenerators>` node. The `<Function>` node recognizes the following parameters:
        - `method`: ***string, optional***, the name of the `<DataGenerator>` from which this value should be taken.
      - `<multiplier>`: ***float***, Multiplies any value obtained by this parameter by the given value.
        Default: 1
    - `<Function>`: ***string***, indicates this value should be taken from a Python function, as described in the `<DataGenerators>` node. The `<Function>` node recognizes the following parameters:
      - `method`: ***string, optional***, the name of the `<DataGenerator>` from which this value should be taken.
    - `<multiplier>`: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1
- `<minimum>`: provides the minimum value at which this component can act, in units of the indicated resource.

  This value can be taken from any *one* of the sources as subnodes (described below): `<fixed_value>`, `<sweep_values>`, `<opt_bounds>`, `<variable>`, `<ARMA>`, `<Function>`, `<ROM>`. The `<minimum>` node recognizes the following parameters:
    - `resource`: ***string, optional***, indicates the resource that defines the minimum activity level for this component, as with the component's capacity.

  The `<minimum>` node recognizes the following subnodes:
    - `<fixed_value>`: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The `<fixed_value>` node recognizes the following parameters:
      - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
    - `<sweep_values>`: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The `<sweep_values>` node recognizes the following parameters:
      - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
    - `<opt_bounds>`: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The `<opt_bounds>` node recognizes the following parameters:

    - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- `<variable>`: ***string***, the name of the variable from inner RAVEN variables.
- `<ARMA>`: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the `<DataGenerator>` node. The `<ARMA>` node recognizes the following parameters:
    - `variable`: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
- `<ROM>`: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The `<ROM>` node recognizes the following parameters:
    - `rom`: ***string, required***, indicates which DataGenerator ROM should be used for this value.
    - `variable`: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

  The `<ROM>` node recognizes the following subnodes:
    - `<input>`: designates the source of one of the inputs to the ROM.

      This value can be taken from any *one* of the sources as subnodes (described below): `<fixed_value>`, `<sweep_values>`, `<opt_bounds>`, `<variable>`, `<ARMA>`, `<Function>`. The `<input>` node recognizes the following parameters:
        - `name`: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

      The `<input>` node recognizes the following subnodes:
        - `<fixed_value>`: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The `<fixed_value>` node recognizes the following parameters:
            - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
        - `<sweep_values>`: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The `<sweep_values>` node recognizes the following parameters:
            - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
        - `<opt_bounds>`: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The `<opt_bounds>` node recognizes the following parameters:

        - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
      - <variable>: ***string***, the name of the variable from inner RAVEN variables.
      - <ARMA>: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the <DataGenerator> node. The <ARMA> node recognizes the following parameters:
        - variable: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
      - <Function>: ***string***, indicates this value should be taken from a Python function, as described in the <DataGenerators> node. The <Function> node recognizes the following parameters:
        - method: ***string, optional***, the name of the <DataGenerator> from which this value should be taken.
      - <multiplier>: ***float***, Multiplies any value obtained by this parameter by the given value.
        Default: 1
    - <Function>: ***string***, indicates this value should be taken from a Python function, as described in the <DataGenerators> node. The <Function> node recognizes the following parameters:
      - method: ***string, optional***, the name of the <DataGenerator> from which this value should be taken.
    - <multiplier>: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1

- <economics>: this node is where all the economic information about this component is placed.

  The <economics> node recognizes the following subnodes:

  - <lifetime>: ***integer***, indicates the number of *cycles* (often *years*) this unit is expected to operate before replacement. Replacement is represented as overnight capital cost in the year the component is replaced.
  - <CashFlow>: node for defining a CashFlow for a particular Component. This HERON CashFlow will be used to generate a TEAL CashFlow from RAVEN's TEAL plugin. Note a CashFlow generally takes the form $C = \alpha \left(\frac{D}{D'}\right)^x$, aggregated depending on the type. For more information, see the TEAL plugin for RAVEN. The <CashFlow> node recognizes the following parameters:

- `name`: ***string, required***, the name by which this CashFlow will be identified as part of this component. The general name is prefixed by the component name, such as ComponentName|CashFlowName.
- `type`: ***[one-time, repeating], required***, the type of CashFlow to calculate. `'('` one-time) is suitable for capital expenditure CashFlows, while `'('` repeating) is used for repeating costs such as operations and maintenance (fixed or variable), market sales, or similar.
- `taxable`: ***[True, Yes, 1, False, No, 0, t, y, 1, f, n, 0], required***, determines whether this CashFlow is taxed every cycle.
- `inflation`: ***string, required***, determines how inflation affects this CashFlow every cycle. See the CashFlow submodule of RAVEN.
- `mult_target`: ***[True, Yes, 1, False, No, 0, t, y, 1, f, n, 0], required***, indicates whether this parameter should be a target of the multiplication factor for NPV matching analyses.
- `period`: ***[hour, year], optional***, for a `<CashFlow>` with `type` `'repeating'`, indicates whether the CashFlow repeats every time step (`'hour'`) or every cycle (`'year'`)). Generally, CashFlows such as fixed operations and maintenance costs are per-cycle, whereas variable costs such as fuel and maintenance as well as sales are repeated every time step.

The `<CashFlow>` node recognizes the following subnodes:

- `<driver>`: indicates the main driver for this CashFlow, such as the number of units sold or the size of the constructed unit. Corresponds to D in the CashFlow equation.

  This value can be taken from any *one* of the sources as subnodes (described below): `<fixed_value>`, `<sweep_values>`, `<opt_bounds>`, `<variable>`, `<ARMA>`, `<Function>`, `<ROM>`, `<activity>`.
  The `<driver>` node recognizes the following subnodes:
  - `<fixed_value>`: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The `<fixed_value>` node recognizes the following parameters:
    - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
  - `<sweep_values>`: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The `<sweep_values>` node recognizes the following parameters:
    - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
  - `<opt_bounds>`: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The

<opt_bounds> node recognizes the following parameters:

   - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.

- **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
- **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
  - variable: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
- **<ROM>**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **<ROM>** node recognizes the following parameters:
  - rom: ***string, required***, indicates which DataGenerator ROM should be used for this value.
  - variable: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

  The **<ROM>** node recognizes the following subnodes:
  - **<input>**: designates the source of one of the inputs to the ROM.

    This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**. The **<input>** node recognizes the following parameters:
    - name: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

    The **<input>** node recognizes the following subnodes:
    - **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
      - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
    - **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
      - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
    - **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner

workflow. The **<opt_bounds>** node recognizes the following parameters:

            - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.

        - **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
        - **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
            - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
        - **<Function>**: ***string***, indicates this value should be taken from a Python function, as described in the **<DataGenerators>** node. The **<Function>** node recognizes the following parameters:
            - **method**: ***string, optional***, the name of the **<DataGenerator>** from which this value should be taken.
        - **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
          Default: 1
    - **<Function>**: ***string***, indicates this value should be taken from a Python function, as described in the **<DataGenerators>** node. The **<Function>** node recognizes the following parameters:
        - **method**: ***string, optional***, the name of the **<DataGenerator>** from which this value should be taken.
    - **<activity>**: ***string***, indicates that this value will be taken from the dispatched activity of this component. The value of this node should be the name of a resource that this component either produces or consumes. Note the sign of the activity by default will be negative for consumed resources and positive for produced resources. The **<activity>** node recognizes the following parameters:
        - **tracking**: ***string, optional***, Some kinds of components have multiple tracking variables. This attribute specifies which tracking variable is desired. Options are only ”production” for Production and Demand units, and [level, charge, discharge] for Storage units. Default is the first entry listed.
    - **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1
- **<reference_price>**: indicates the cash value of the reference number of units sold. corresponds to α in the CashFlow equation. If **<reference_driver>** is

1, then this is the price-per-unit for the CashFlow.

This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**, **<ROM>**, **<activity>**.
The **<reference_price>** node recognizes the following subnodes:

- **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
- **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
  - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
- **<ROM>**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **<ROM>** node recognizes the following parameters:
  - **rom**: ***string, required***, indicates which DataGenerator ROM should be used for this value.
  - **variable**: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

  The **<ROM>** node recognizes the following subnodes:
  - **<input>**: designates the source of one of the inputs to the ROM.

    This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**,

<ARMA>, **<Function>**. The **<input>** node recognizes the following parameters:

- name: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

The **<input>** node recognizes the following subnodes:

- **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
  - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
  - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
  - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.
- **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
- **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
  - variable: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
- **<Function>**: ***string***, indicates this value should be taken from a Python function, as described in the **<DataGenerators>** node. The **<Function>** node recognizes the following parameters:
  - method: ***string, optional***, the name of the **<DataGenerator>** from which this value should be taken.
- **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
  Default: 1

- **<Function>**: ***string***, indicates this value should be taken from a Python function, as described in the **<DataGenerators>** node. The **<Function>** node recognizes the following parameters:

  - method: ***string, optional***, the name of the **<DataGenerator>** from which this value should be taken.

- **<activity>**: ***string***, indicates that this value will be taken from the dispatched activity of this component. The value of this node should be the name of a resource that this component either produces or consumes. Note the sign of the activity by default will be negative for consumed resources and positive for produced resources. The **<activity>** node recognizes the following parameters:

  - tracking: ***string, optional***, Some kinds of components have multiple tracking variables. This attribute specifies which tracking variable is desired. Options are only "production" for Production and Demand units, and [level, charge, discharge] for Storage units. Default is the first entry listed.

- **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
  Default: 1

- **<reference_driver>**: determines the number of units sold to which the **<reference_pri**
refers. Corresponds to $\prime D$ in the CashFlow equation.

  This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**, **<ROM>**, **<activity>**.
  The **<reference_driver>** node recognizes the following subnodes:

  - **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:

    - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.

  - **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:

    - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.

  - **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:

    - debug_value: ***float, optional***, provides a value to use for this entry when in Case mode has "debug" enabled.

  - **<variable>**: ***string***, the name of the variable from inner RAVEN variables.

- **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
  - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
- **<ROM>**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **<ROM>** node recognizes the following parameters:
  - **rom**: ***string, required***, indicates which DataGenerator ROM should be used for this value.
  - **variable**: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

  The **<ROM>** node recognizes the following subnodes:
  - **<input>**: designates the source of one of the inputs to the ROM.

    This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**. The **<input>** node recognizes the following parameters:
    - **name**: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

    The **<input>** node recognizes the following subnodes:
    - **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
      - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
      - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
      - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.

        - **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
        - **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
            - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
        - **<Function>**: ***string***, indicates this value should be taken from a Python function, as described in the **<DataGenerators>** node. The **<Function>** node recognizes the following parameters:
            - **method**: ***string, optional***, the name of the **<DataGenerator>** from which this value should be taken.
        - **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
          Default: 1
    - **<Function>**: ***string***, indicates this value should be taken from a Python function, as described in the **<DataGenerators>** node. The **<Function>** node recognizes the following parameters:
        - **method**: ***string, optional***, the name of the **<DataGenerator>** from which this value should be taken.
    - **<activity>**: ***string***, indicates that this value will be taken from the dispatched activity of this component. The value of this node should be the name of a resource that this component either produces or consumes. Note the sign of the activity by default will be negative for consumed resources and positive for produced resources. The **<activity>** node recognizes the following parameters:
        - **tracking**: ***string, optional***, Some kinds of components have multiple tracking variables. This attribute specifies which tracking variable is desired. Options are only ”production” for Production and Demand units, and [level, charge, discharge] for Storage units. Default is the first entry listed.
    - **<multiplier>**: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1
- **<scaling_factor_x>**: determines the scaling factor for this CashFlow. Corresponds to x in the CashFlow equation. If x is less than one, the per-unit price decreases as the units sold increases above the **<reference_driver>**, and vice versa.

  This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**, **<ROM>**, **<activity>**.

The **<scaling_factor_x>** node recognizes the following subnodes:

- **<fixed_value>**: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The **<fixed_value>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
- **<sweep_values>**: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The **<sweep_values>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
- **<opt_bounds>**: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The **<opt_bounds>** node recognizes the following parameters:
  - **debug_value**: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
- **<variable>**: ***string***, the name of the variable from inner RAVEN variables.
- **<ARMA>**: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the **<DataGenerator>** node. The **<ARMA>** node recognizes the following parameters:
  - **variable**: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
- **<ROM>**: ***string***, indicates that this value will be taken from a RAVEN-trained ROM. The **<ROM>** node recognizes the following parameters:
  - **rom**: ***string, required***, indicates which DataGenerator ROM should be used for this value.
  - **variable**: ***string, required***, indicates the variable output of the ROM from which this value should be taken.

  The **<ROM>** node recognizes the following subnodes:
  - **<input>**: designates the source of one of the inputs to the ROM.

    This value can be taken from any *one* of the sources as subnodes (described below): **<fixed_value>**, **<sweep_values>**, **<opt_bounds>**, **<variable>**, **<ARMA>**, **<Function>**. The **<input>** node recognizes the following parameters:

    - `name`: ***string, required***, name of the variable to pass into the ROM evaluation. Determined by the ROM training.

  The `<input>` node recognizes the following subnodes:

    - `<fixed_value>`: ***float***, indicates this value should be fixed in the outer run, and act as a constant in the inner workflow. The `<fixed_value>` node recognizes the following parameters:
        - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - `<sweep_values>`: ***comma-separated floats***, indicates this value should be parametrically swept in the outer run, while acting as a constant in the inner workflow. The `<sweep_values>` node recognizes the following parameters:
        - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - `<opt_bounds>`: ***comma-separated floats***, indicates this value should be optimized in the outer run, while acting as a constant in the inner workflow. The `<opt_bounds>` node recognizes the following parameters:
        - `debug_value`: ***float, optional***, provides a value to use for this entry when in Case mode has ”debug” enabled.
    - `<variable>`: ***string***, the name of the variable from inner RAVEN variables.
    - `<ARMA>`: ***string***, indicates that this value will be taken from synthetically-generated signals, which will be provided to the dispatch at run time by RAVEN from trained models. The value of this node should be the name of a synthetic history generator in the `<DataGenerator>` node. The `<ARMA>` node recognizes the following parameters:
        - `variable`: ***string, optional***, indicates which variable coming from the synthetic histories this value should be taken from.
    - `<Function>`: ***string***, indicates this value should be taken from a Python function, as described in the `<DataGenerators>` node. The `<Function>` node recognizes the following parameters:
        - `method`: ***string, optional***, the name of the `<DataGenerator>` from which this value should be taken.
    - `<multiplier>`: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1

- `<Function>`: ***string***, indicates this value should be taken from a Python function, as described in the `<DataGenerators>` node. The `<Function>` node recognizes the following parameters:

        - `method`: ***string, optional***, the name of the `<DataGenerator>` from which this value should be taken.
    - `<activity>`: ***string***, indicates that this value will be taken from the dispatched activity of this component. The value of this node should be the name of a resource that this component either produces or consumes. Note the sign of the activity by default will be negative for consumed resources and positive for produced resources. The `<activity>` node recognizes the following parameters:
        - `tracking`: ***string, optional***, Some kinds of components have multiple tracking variables. This attribute specifies which tracking variable is desired. Options are only ”production” for Production and Demand units, and [level, charge, discharge] for Storage units. Default is the first entry listed.
    - `<multiplier>`: ***float***, Multiplies any value obtained by this parameter by the given value.
      Default: 1
- `<depreciate>`: ***integer***, – no description yet –

6 Placeholders

6.1 ARMA

This data source is a source of synthetically-generated histories trained by RAVEN. The RAVEN ARMA ROM should be trained and serialized before using it in HERON. The text of this node indicates the location of the serialized ROM. This location is usually relative with respect to the HERON XML input file; however, a full absolute path can be used, or the path can be prepended with "%HERON%" to be relative to the installation directory of HERON.

The `<ARMA>` node recognizes the following parameters:

- `name`: ***string, required***, identifier for this data source in HERON and in the HERON input file.
- `variable`: ***comma-separated strings, required***, provides the names of the variables from the synthetic history generators that will be used in this analysis.
- `evalMode`: ***string, optional***, desired sampling mode for the ARMA. See the RAVEN manual for options.
  Default: clustered

6.2 Function

This data source is a custom Python function to provide derived values. Python functions have access to the variables within the dispatcher. The text of this node indicates the location of the python file. This location is usually relative with respect to the HERON XML input file; however, a full absolute path can be used, or the path can be prepended with "%HERON%" to be relative to the installation directory of HERON.

The `<Function>` node recognizes the following parameters:

- `name`: ***string, required***, identifier for this data source in HERON and in the HERON input file.

7 External Code Integration

HERON can exchange data with other Integrated Energy Systems (IES) codes to conduct technical or economic analyses for various electricity market structures. More information about integrating HERON with other IES software is here: `https://ies.inl.gov/SitePages/FORCE.aspx`

The integration between HERON and other IES tools is still a work in progress. Currently, HERON can communicate with the following codes

7.1 HYBRID

The HYBRID repository contains a collection of models representing the physical dynamics of various integrated energy systems and processes. HERON has the capability to load the economic information about the components of the grid system automatically from HYBRID. More information about HYBRID is here: `https://github.com/idaholab/HYBRID`

An example that demonstrates this capability can be found at:

```
/HERON/tests/integration_tests/mechanics/hybrid_load/
```

The Python script, that auto-loads the needed economic information from HYBRID to HERON, is named `hybrid2heron_economic.py`. This script can be found at:

```
/HERON/src/Hybrid2Heron/
```

The `hybrid2heron_economic.py` script takes one command-line argument, which is the path of the initial HERON input XML file (or pre-input file) before loading any information from HYBRID

For example, the terminal command looks like this:

```
python hybrid2heron_economic.py pre_heron_input.xml
```

A new HERON input XML file, `heron_input.xml`, is generated with all the subnodes under the **`<economics>`** node loaded from the HYBRID text files. More details about the initial HERON input XML file, the generated (loaded) input XML file and other files at the `/HERON/tests/integration_tests/mechanics/hybrid_load/` test are discussed in detail in the following subsections.

7.1.1 The initial HERON input XML file

The initial HERON XML file, `pre_heron_input.xml`, structure should be similar to the typical HERON input XML file and must have

- A `<Components>` node:
- At least one `<Component>` sub-node under the `<Components>` node such as:

  ```
  <Component name="component_name"> </Component>
  ```

- An empty `<economics>` node under the `<Component>` node with the path to the HYBRID text file that includes the needed information as follows:

  ```
  <economics src="path/to/HYBRID/file"> </economics>
  ```

If the `<economics>` node is not empty, it will remain unaltered when creating the final HERON XML file, `heron_input.xml`.

An example of the initial HERON XML file, `pre_heron_input.xml`, is located at `/HERON/tests/integration_tests/mechanics/hybrid_load/`. The `<Components>` node at the `pre_heron_input.xml` looks like this:

```
<Components>
  <Component name="source">
    <!--Other component subnodes-->
    <economics src="Costs/source/source.toml"></economics>
  </Component>
</Components>
```

In this example, the economic information of the component `'source'` would be loaded from a text file whose path is `/Costs/source/source.toml`. Similarly, for any other component, the economic information can be provided from other text files.

7.1.2 The HYBRID text files

The HYBRID text files are expected to have extensions such as .toml or .txt or .rtf and are located at: `/HERON/tests/integration_tests/mechanics/hybrid_load/Costs/sink/sink.toml` and `/HERON/tests/integration_tests/mechanics/hybrid_load/Costs/source/source.toml`.

The HYBRID files do not have to be in the same folder with the pre_heron_input.xml as long as the value of the **'src'** parameter at the **<economics>** node is the path to the corresponding HYBRID text file. The HYBRID text file structure looks like this:

```
Lifetime = 30 #years
VOM = 0 # Just a placeholder to make sure it passes through
Activity = a
```

Each line, in the HYBRID text file, includes a variable name and its value plus a comment (if necessary). Any comments must start with the # sign. The data should be appropriately autoloaded from HYBRID text file to HERON XML file even if the text file includes additional irrelevant variables or additional comments at the top or the bottom of the file.

7.1.3 The generated HERON Input XML file

The generated input file, heron_input.xml, includes:

- All the information that was in the initial HERON input file, pre_heron_input.xml
- All the relevant variables from the HYBRID text files.
- Default values for additional variables that are found neither at the pre_heron_input.xml nor at the HYBRID text files but are required by HERON to make sure that the input XML file is complete, and no required nodes or parameters are missing. The comments (warnings) inside the heron_input.xml tell the user if the values of specific variables are not provided by HYBRID, and if default values are assigned instead. The user should review these comments/warnings.

7.1.4 HYBRID and HERON keywords

The HYBRID keywords or the HYBRID variables that the hybrid2heron_economic.py code can identify to create the corresponding HERON nodes are listed in the CSV file, HYBRID_HERON_keywords.csv, which is located at: /HERON/src/Hybrid2Heron. Understanding this CSV file is essential, especially if the user plans to add more HYBRID variables or modify them. The CSV file, HYBRID_HERON_keywords.csv includes the following columns:

- **HYBRID Keyword**: This column lists all the HYBRID variables' names that the Python script, hybrid2heron_economic.py, can identify. The variables' names in the HYBRID text files must be a subset of the HYBRID variables' names at the HYBRID_HERON_keywords.csv. Otherwise, the user can either change the variables' names in the HYBRID_HERON_keywords.csv file or in the HYBRID text files.

- **Description**: The description or the definition of each HYBRID variable
- **HERON (Node or Parameter)**: This column specifies if the HYBRID variable is corresponding to a HERON node or a node parameter in the HERON input XML file. *N* refers to a node while *P* refers to a parameter.
- **HERON Node**, **HERON Subnode** and **HERON Subsubnode**: These three columns specify the location of the HERON node corresponding to the HYBRID variable. For example, the HYBRID variable `"Activity"` corresponds the sub-sub-node `<activity>` under the `<driver>` sub-node under the `<CashFlow>` node as follows:

```
<CashFlow inflation="none" mult_target="FALSE" name="VOM"
   taxable="TRUE" type="repeating">
    <driver>
        <activity>a</activity>
    </driver>
</CashFlow>
```

  Also, the HYBRID variable `"VOM_inflation"` corresponds to a *parameter(P)* or an attribute that is called `'inflation'` at the node `<CashFlow>` under the `<economics>` node as illustrated in the `<economics>` node (above).

- **Belong to same node**: This column is intended to determine the list of sub-nodes or parameters that belong to the same node. We consider four primary nodes under the `<economics>` node, which are the component lifetime plus three types of cash flows: The Capital Expenditures (CAPEX) cash flow, the Fixed Operation and Maintenance (FOM) cash flow, and the Variable Operation and Maintenance (VOM) cash flow. These four primary nodes are listed under the `<economics>` node at the HERON input XML file are as follows:

```
<economics>
  <lifetime>30</lifetime>
  <CashFlow inflation="none" mult_target="FALSE"
     name="capex" taxable="TRUE" type="one-time"></CashFlow>
  <CashFlow inflation="none" mult_target="FALSE" name="FOM"
     taxable="TRUE" type="repeating"></CashFlow>
  <CashFlow inflation="none" mult_target="FALSE" name="VOM"
     taxable="TRUE" type="repeating"></CashFlow>
</economics>
```

  The numerical values under the `"Belong to same node"` column are either `0` or `1` or `2` or `3` corresponding to the nodes `<lifetime>`, `<CashFlow name="capex">`, `<CashFlow name="FOM">`, `<CashFlow name="VOM">` respectively.

  For example, since the HYBRID variable `"capex_inflation"` corresponds to the parameter, `'inflation'`, under the node `<CashFlow name="capex">`, the corresponding numerical value under the `"Belong to same node"` is `"1"`.

Similarly, the `"Amortization_lifetime"` HYBRID variable corresponds to the HERON sub-node **`<depreciate>`** under *all* the three cash flow nodes, the corresponding numerical value under the `"Belong to same node"` is `"1,2,3"`.

Note that we consider three types only of cash flows, but the user can add additional cash flows to the `HYBRID_HERON_keywords.csv` file, if needed.

- **Required if HYBRID keyword?**: This column determines if the HYBRID variable needs to be included when building the HERON input XML file even if this HYBRID variable is not provided by the HYBRID text files.

  For example, the HYBRID variable, `"Activity"`, will be included if the `"VOM"` cash flow is present since the node **`<CashFlow name="VOM">`** will be incomplete if the **`<activity>`** sub-node is missing. Therefore, for the HYBRID variable, `"Activity"`, the corresponding value under the `"Required if HYBRID keyword?"` column is `"VOM"`

- **Default value for the required variable**: This column assigns a default value for any HYBRID variable whose value is not provided by the HYBRID text files if this HYBRID variable is required (see the column `"Required if HYBRID keyword?"`). For example, the default value of the HYBRID variable, `"Activity"`, if required, is **`'electricity'`**

Idaho National Laboratory