BuildStock Documentation

Release 2.2.2

NREL

Feb 24, 2020

Contents:

CHAPTER 1

Tutorial

1.1 Installation

1.1.1 Download the ResStock repository

Go to the releases page on GitHub, select a release, and either `git clone` or download a zip of the repository. The `master` branch is under active development, so we recommend using the latest release for production runs.

1.1.2 Developer instructions

If you will be developing residential measures and testing residential building models, see the *Advanced Tutorial*. If you are a developer, make sure that you have checked out the `master` branch of the repository.

1.2 Set Up the Analysis Project

At the top level of the ResStock repository you just downloaded, you will see two analysis project folders:

- project_multifamily_beta
- project_testing

1.2.1 OpenStudio Measures

Simulation Controls

Using this measure you can set the simulation timesteps per hour, the run period begin month/day and end month/day, and the calendar year (for start day of week). By default the simulations use a 10-min timestep (i.e., the number of timesteps per hour is 6), start on January 1, end on December 31, and run with a calendar year of 2007 (start day of week is Monday). If you are running simulations using AMY weather files, the value entered for calendar year will not be used; it will be overridden by the actual year found in the AMY weather file.

Build Existing Model

This measure creates the baseline scenario. It incrementally applies OpenStudio measures (located in the `resources` directory, which should be at the same level as your project directory) to create residential building models. Set the following inputs:

Building ID – Max This sets the number of simulations to run in the baseline and each upgrade case. For this tutorial I am going to set this to 1000. Most analyses will require more, but we're going to keep the total number small for simulation time and cost.

Number of Buildings Represented The total number of buildings this sampling is meant to represent. This sets the weighting factors. For the U.S. single-family detached housing stock, this is 80 million homes.

Sample Weight of Simulation The number of buildings each simulation represents. Total number of buildings / Number of simulations. This argument is optional (it is only needed for running simulations on NREL HPC), so you can leave it blank.

Downselect Logic Logic that specifies the subset of the building stock to be considered in the analysis. Specify one or more `parameter|option` as found in the `resources/options_lookup.tsv`. (This uses the same syntax as the *Apply Upgrade* measure.) For example, if you wanted to only simulate California homes you could enter `Location Region|CR11` in this field (CR refers to "Custom Region", which is based on RECS 2009 reportable domains aggregated into groups with similar climates; see the entire custom region map). Datapoints that are excluded from the downselect logic will result in "completed invalid workflow". Note that the **Building ID - Max** input refers to the number of datapoints *before* downselection, not after. This means that the number of datapoints remaining after downselection would be somewhere between zero (i.e., no datapoints matched the downselect logic) and **Building ID - Max** (i.e., all datapoints matched the downselect logic).

Measures to Ignore **INTENDED FOR ADVANCED USERS/WORKFLOW DEVELOPERS ONLY.** Measures to exclude from the OpenStudio Workflow specified by listing one or more measure directories separated by '|'. Core ResStock measures cannot be ignored (the Build Existing Model measure will fail).

Note: **Manual Sampling**: To run the sampling script yourself, from the command line execute, e.g. `ruby resources/run_sampling.rb -p project_multifamily_beta -n 10000 -o buildstock.csv`, and a file `buildstock.csv` will be created in the `resources` directory.

If a custom `buildstock.csv` file is located in a project's `housing_characteristics` directory when you run the project, it will automatically be used to generate simulations. If it's not found, the `run_sampling.rb` script will be run automatically on OpenStudio-Server to create one. You'll also want to make sure that the number of buildings in the sampling csv file matches the max value for the Building ID argument in the Build Existing Model, as that tells OpenStudio how many datapoints to run. (For each datapoint, the measure will then look up its building description from the sampling csv.)

You can use this manual sampling process to downselect which simulations you want to run. For example, you can use the command above to generate a `buildstock.csv` for the entire U.S. and then open up this file in Excel and delete all of the rows that you don't want to simulate (e.g., all rows that aren't in New York). Keep in mind that if you do this, you will need to re-enumerate the "Building" column as "1" through the number of rows.

Apply Upgrade

Each "Apply Upgrade" measure defines an upgrade scenario. An upgrade scenario is a collection of options exercised with some logic and costs applied. In the simplest case, we apply the new option to all houses. The available upgrade options are in `resources/options_lookup.tsv` in your git repository.

For this example, we will upgrade all windows by applying the `Windows|Low-E, Triple, Non-metal, Air, L-Gain` option to all houses across the country. We do this by entering that in the **Option 1** box on the Apply Upgrade measure. Also, we'll give the upgrade scenario a name: "Triple-Pane Windows" and a cost of $40/ft^2

of window area by entering the number in **Option 1 Cost Value** and selecting "Window Area (ft^2)" for **Option 1 Cost Multiplier**.

Like the **downselect logic**, excluded datapoints (i.e., datapoints for which the upgrade does not apply) will result in "completed invalid workflow". For a full explanation of how to set up the options and logic surrounding them, see *Upgrade Scenario Configuration*.

1.2.2 Reporting Measures

Scroll down to the bottom on the Measures Selection tab, and you will see the **Reporting Measures** section. This section is where you can request timeseries data and utility bills for the analysis. In general, reporting measures process data after the simulation has finished and produced results. As a note, make sure that the **Timeseries CSV Export** and **Utility Bill Calculations** measures are placed before the **Server Directory Cleanup** measure.

Simulation Output Report

Leave this alone if you do not want to report annual totals for end use subcategories. Select **Include End Use Subcategories** if you want to report them. See below for a listing of available end use subcategories.

Timeseries CSV Export

If you do not need the timeseries data for your simulations, you can skip this measure to save disk space. Otherwise, one csv file per datapoint will be written containing end use timeseries data for their model.

End uses include:

- total site energy [MBtu]
- net site energy [MBtu]
- total site [electric/gas/oil/propane/wood] [kWh/therm/MBtu/MBtu/MBtu]
- net site [electric] [kWh]
- heating [electric/gas/oil/propane/wood] [kWh/therm/MBtu/MBtu/MBtu]
- cooling [kWh]
- central system heating [electric/gas/oil/propane] [kWh/therm/MBtu/MBtu]
- central system cooling [electric] [kWh]
- interior lighting [kWh]
- exterior lighting [kWh]
- exterior holiday lighting [kWh]
- garage lighting [kWh]
- interior equipment [electric/gas/propane] [kWh/therm/MBtu/MBtu]
- fans heating [kWh]
- fans cooling [kWh]
- pumps heating [kWh]
- pumps cooling [kWh]
- central system pumps heating [electric] [kWh]

- central system pumps cooling [electric] [kWh]
- water heating [electric/gas/oil/propane] [kWh/therm/MBtu/MBtu]
- pv [kWh]

Reporting Frequency The timeseries data will be reported at hourly intervals unless otherwise specified. Alternative reporting frequencies include:

- Timestep
- Daily
- Monthly
- Runperiod

Setting the reporting frequency to 'Timestep' will give you interval output equal to the zone timestep set by the *Simulation Controls* measure. Thus, this measure will produce 10-min interval output when you select 'Timestep' and leave the *Simulation Controls* measure at its default settings.

Include End Use Subcategories Select this to include end use subcategories. The default is to not include end use subcategories. End use subcategories include:

- refrigerator [kWh]
- clothes washer [kWh]
- clothes dryer [electric/gas/propane] [kWh/therm/MBtu]
- cooking range [electric/gas/propane] [kWh/therm/MBtu]
- dishwasher [kWh]
- plug loads [kWh]
- house fan [kWh]
- range fan [kWh]
- bath fan [kWh]
- ceiling fan [kWh]
- extra refrigerator [kWh]
- freezer [kWh]
- pool heater [electric/gas] [kWh/therm]
- pool pump [kWh]
- hot tub heater [electric/gas] [kWh/therm]
- hot tub pump [kWh]
- gas grill [therm]
- gas lighting [therm]
- gas fireplace [therm]
- well pump [kWh]
- hot water recirculation pump [kWh]
- vehicle [kWh]

Output Variables If you choose to report any output variables (e.g., "Zone Air Temperature" or "Site Outdoor Air Humidity Ratio"), enter a comma-separated list of output variable names. A list of available output variables can be viewed in EnergyPlus's `.rdd` file.

Utility Bill Calculations

This measure is currently under construction.

1.3 Run the Project

See the buildstockbatch documentation for information on running projects.

1.4 Conclusion

Congratulations, you have now completed your first ResStock analysis. See the other sections in this documentation for more advanced topics.

CHAPTER 2

Advanced Tutorial

This advanced tutorial describes the process for developing residential measures and testing residential building models. Reasons for wanting to develop residential measures include: customizing any of the existing residential modeling algorithms or adding new technology models.

At this point in the tutorial, it is assumed that you have checked out a new branch that is up-to-date with the **master branch** of the OpenStudio-BuildStock repository. Optionally, you may have created a new project folder (i.e., copied an existing project folder) and modified the set of tsv files in its `housing_characteristics` folder.

If your changes are intended to be merged into the master branch of the OpenStudio-BuildStock repository, a pull request review is required.

2.1 Modifying Probability Distributions

This section provides a description of the housing characteristics and their dependencies and options.

A particular building within the building stock has a set of characteristics (e.g., level of wall insulation, type of lighting, vintage, and a variety of different schedules). Each housing characteristic corresponds to a tab-separated value (tsv) file with the extension *.tsv*. These housing characteristics files are found in the `<project_folder>/housing_characteristics` directory. A housing characteristic defines the probability mass function (PMF) of that characteristic in the building stock.

$Pr(X = A_i) = P(A_i) > 0 \quad and \quad \sum_{A_i \in S_A} P(A_i) = 1 \quad for \quad i = 1 : n$

When sampling a discrete random variable X to create a representative building, X takes a particular **Option** A_i. All possible options are collected in the set $S_A = \{A_0, A_1, ..., A_n\}$ and is size n. Since these are probabilities, the entries $P(A_i)$ must be greater than 0 and the probability of all possible options must sum to 1.

For example, a set of options for a building's vintage (when the building was built) may be the following:

$S_A = < 1950, 1950s, 1960s, 1970s, 1980s, 1990s, 2000s.$

Then the probability mass function may look like the following:

A_i	<1950	1950s	1960s	1970s	1980s	1990s	2000s
$P(X = A_i)$	0.020	0.060	0.090	0.230	0.370	0.130	0.090

Where the probability of a building having a given vintage in this example is

- 2% built before 1950,
- 6% in the 1950s,
- 9% in the 1960s,
- 23% in the 1970s,
- 37% in the 1980s,
- 13% in the 1900s, and
- 9% in the 2000s.

However, housing characteristics can have a **Dependency**, B_i, to another housing characteristic. All possible values of the dependency are collected in the set $S_B = B_0, B_1, ...B_m$ which is size m. If the **Option** of interest A_j and the **Dependency** B_i is known to have occurred when sampling X in the creation of a representative building, then conditional probability of A_j given B_i is usually written $P(A_j|B_i) = P_{B_i}(A_j)$.

Using the example from before, the PMF of the vintage depends on location of the particular building stock (which is represented by EPW weather files). In this example the vintage housing characteristic is examined. The first three lines in the `<project_folder>/housing_characteristics/Vintage.tsv` are shown in the table below.

	Location EPW (S_B)	<1950	1950s	1960s	1970s	1980s	1990s	2000s
$P(B_0\|A_j)$	USA_FL_Key.West.Intl.AP.722010_TMY3.epw	0.02	0.06	0.09	0.23	0.37	0.13	0.09
$P(B_1\|A_j)$	USA_FL_Miami.Intl.AP.722020_TMY3.epw	0.05	0.13	0.13	0.18	0.17	0.18	0.16

The vintage is dependent on the EPW location. The vintage discrete PMF that uses the Key West International Airport weather file, B_0, is defined by the following distribution:

- 2% built before 1950,
- 6% in the 1950s,
- 9% in the 1960s,
- 23% in the 1970s,
- 37% in the 1980s,
- 13% in the 1900s, and
- 9% in the 2000s.

While the vintage PMF that uses the Miami International Airport weather file, B_1 is defined by the following distribution:

- 5% built before 1950,
- 13% in the 1950s,
- 9% in the 1960s,
- 13% in the 1970s,
- 18% in the 1980s,
- 17% in the 1900s, and

- 18% in the 2000s.

The **Options** can correspond to a Measure in OpenStudio or can be used as a **Dependency** for other housing characteristics. For the list of available options for a given housing characteristic, see the `resources/options_lookup.tsv` file. In this file the "Parameter Name" corresponds to the housing characteristic, the "Option Name" corresponds to an available option for the housing characteristic, the "Measure Dir" corresponds to the OpenStudio Measure being used, and the following columns correspond to different arguments needed by the OpenStudio Measure. Each option used in the housing characteristics tsv files must be in this `resources/options_lookup.tsv`. These options can be modified by the user to model their particular building stock.

If adding or renaming any housing characteristics tsv files, refer to the refresh-outputs section for instructions on how to get the sampled options to show up in results files.

2.2 Writing Housing Characteristics

This section provides a description of the standard format for the housing characteristics. In order to stop recommitting entire files and to keep differences easy to read during the review process, a standard format for writing housing characteristics has been created. All housing characteristics shall follow these format guidelines:

2.2.1 Guidelines

1. All lines have the line ending characters '\r\n' (i.e., "crlf" or "carriage return, line feed").
2. All non-dependency columns have format '%.6f'.
3. Comment lines in the housing characteristics file, indicated by the "#" symbol, can be added.

2.3 Installer Setup

After you have downloaded the OpenStudio installer, you will want to install Ruby (2.2.4). This will allow you to execute rake tasks contained in the Rakefile. Follow the instructions below for *Windows Setup* or *Mac Setup*.

2.3.1 Windows Setup

1. Install Ruby (2.2.4). Follow the installation instructions here ("Optional - Install Ruby").
2. Run `gem install bundler -v 1.17.1`.

Note: If you get an error, you may have to issue the following: `gem sources -r https://rubygems.org/` followed by `gem sources -a http://rubygems.org/`. If you still get an error, manually update your gem sources list by including a config file named ".gemrc" in your home directory (.e.g, /c/Users/<USERNAME>) with the following contents:

```
---
:backtrace: false
:bulk_threshold: 1000
:sources:
- http://rubygems.org
:update_sources: true
:verbose: true
```

3. Download the DevKit at http://rubyinstaller.org/downloads/ (e.g., DevKit-mingw64-64-4.7.2-20130224-1432-sfx.exe). Choose either the 32-bit or 64-bit version depending on which version of Ruby you installed. Run the installer and extract to a directory (e.g., C:\RubyDevKit). Go to this directory, run `ruby dk.rb init`, modify the config.yml file as needed, and finally run `ruby dk.rb install`.
4. Run `bundle install` from the OpenStudio-BuildStock directory. (If you get an error, the problem may be that `git` is not in your `PATH`.)

2.3.2 Mac Setup

Install Homebrew if you don't have it already.

Run `brew doctor`. It should give you, among other issues, a list of unexpected dylibs that you'll need to move for this to work such as:

```
Unexpected dylibs:
  /usr/local/lib/libcrypto.0.9.8.dylib
  /usr/local/lib/libcrypto.1.0.0.dylib
  /usr/local/lib/libcrypto.dylib
  /usr/local/lib/libklcsagt.dylib
  /usr/local/lib/libklcskca.dylib
  /usr/local/lib/libklcsnagt.dylib
  /usr/local/lib/libklcsrt.dylib
  /usr/local/lib/libklcsstd.dylib
  /usr/local/lib/libklcstr.dylib
  /usr/local/lib/libklmspack.0.1.0.dylib
  /usr/local/lib/libklmspack.0.dylib
  /usr/local/lib/libklmspack.dylib
  /usr/local/lib/libssl.0.9.8.dylib
  /usr/local/lib/libssl.1.0.0.dylib
  /usr/local/lib/libssl.dylib
  /usr/local/lib/libz.1.2.5.dylib
  /usr/local/lib/libz.1.2.6.dylib
  /usr/local/lib/libz.1.dylib
  /usr/local/lib/libz.dylib
```

Highlight and copy the list (without the header "Unexpected dylibs:"). Run the following commands to move them to another location where they won't interfere.

```
mkdir ~/unused_dylibs
pbpaste | xargs -t -I % mv % ~/unused_dylibs
```

Install `rbenv` and required dependencies.

```
brew install openssl libyaml libffi rbenv
```

Initialize `rbenv` by running the command below and following the instructions to add the appropriate things to your `~/.bash_profile`.

```
rbenv init
```

Install the appropriate ruby version.

```
cd path/to/repo
rbenv install `cat .ruby-version`
```

Add the path to the install ruby libraries top the bottom of your `~/.bash_profile`

```
echo "export RUBYLIB=/Applications/OpenStudio-2.9.0/Ruby" >> ~/.bash_profile
echo "export ENERGYPLUS_EXE_PATH=\"/Applications/OpenStudio-2.9.0/EnergyPlus/
↪energyplus-9.2.0\""
```

Install bundler and the libraries that bundler installs.

```
gem install bundler -v 1.17.1
bundle install
```

2.4 Rake Tasks

Once you have completed instructions found in *Installer Setup*, you can then *use the Rakefile* contained at the top level of this repository (Rakefile). You will run rake task(s) for *performing integrity checks on project inputs*.

2.4.1 Using the Rakefile

Run `rake -T` to see the list of possible rake tasks. The `-T` is replaced with the chosen task.

```
$ rake -T
rake integrity_check_all                    # Run tests for integrity_check_all
rake integrity_check_multifamily_beta       # Run tests for integrity_check_...
rake integrity_check_testing                # Run tests for integrity_check_...
rake integrity_check_unit_tests             # Run tests for integrity_check_...
rake test:measures_osw                      # Run tests for measures_osw
rake test:regenerate_osms                   # Run tests for regenerate_osms
rake test:regression_tests                  # Run tests for regression_tests
rake test:unit_tests                        # Run tests for unit_tests
rake update_measures                        # Run tests for update_measures
```

2.4.2 Integrity Checks

Run `rake integrity_check_<project_name>`, where `<project_name>` matches the project you are working with. If no rake task exists for the project you are working with, extend the list of integrity check rake tasks to accommodate your project by copy-pasting and renaming the `integrity_check_multifamily_beta` rake task found in the Rakefile. An example for running a project's integrity checks is given below:

```
$ rake integrity_check_multifamily_beta
Checking for issues with project_multifamily_beta/Location Region...
Checking for issues with project_multifamily_beta/Location EPW...
Checking for issues with project_multifamily_beta/Vintage...
Checking for issues with project_multifamily_beta/Heating Fuel...
Checking for issues with project_multifamily_beta/Usage Level...
...
```

If the integrity check for a given project fails, you will need to update either your tsv files and/or the `resources/options_lookup.tsv` file. See *Options Lookup* for information about the `options_lookup.tsv` file.

2.5 Options Lookup

The `options_lookup.tsv` file, found in the `resources` folder, specifies mappings from sampled options into measure arguments. For example, if the distribution of cooling system types in `HVAC System Cooling.tsv` has `Option=AC, SEER 13` and `Option=AC, SEER 15`, but you want to include a `Option=AC, SEER 17` option, you would add that option as a column in `HVAC System Cooling.tsv` and then create a corresponding row in `options_lookup.tsv`. Updates to this file will allow you to avoid hitting the following types of integrity check errors:

- *Could not find parameter and option*
- *Required argument not provided*

Once you have updated your `options_lookup.tsv` file and all integrity checks are passing, you can move on to updating_projects.

2.5.1 Could not find parameter and option

You do not have a row in `options_lookup.tsv` for a particular option that is sampled.

An example of this error is given below:

```
ERROR: Could not find parameter 'Insulation Wall' and option 'Wood Stud, Uninsulated' in C:/OpenStudio/OpenStudio-BuildStock/resources/options_lookup.tsv.
C:/OpenStudio/OpenStudio-BuildStock/resources/meta_measure.rb:224:in `register_error'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:296:in `block in get_measure_args_from_option_names'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:294:in `each'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:294:in `get_measure_args_from_option_names'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:351:in `block in integrity_check'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:319:in `each'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:319:in `integrity_check'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:237:in `block in <top (required)>'
Tasks: TOP => integrity_check_resstock_national
(See full trace by running task with --trace)
```

2.5.2 Required argument not provided

For the particular option that is sampled, your corresponding measure is missing an argument value assignment.

An example of this error is given below:

```
ERROR: Could not find parameter 'Insulation Wall' and option 'Wood Stud, Uninsulated' in C:/OpenStudio/OpenStudio-BuildStock/resources/options_lookup.tsv.
C:/OpenStudio/OpenStudio-BuildStock/resources/meta_measure.rb:224:in `register_error'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:296:in `block in get_measure_args_from_option_names'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:294:in `each'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:294:in `get_measure_args_from_option_names'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:351:in `block in integrity_check'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:319:in `each'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:319:in `integrity_check'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:237:in `block in <top (required)>'
Tasks: TOP => integrity_check_resstock_national
(See full trace by running task with --trace)
```

2.6 Increasing Upgrade Options

To allow more options per upgrade, increase the value returned by the following method defined in `resources/measures/HPXMLtoOpenStudio/resources/constants.rb`:

```
def self.NumApplyUpgradeOptions
  return 25
end
```

Then run the `update_measures` rake task. See *Rake Tasks* for instructions on how to run rake tasks.

Then you will need to update the outputs section for all PAT projects. See updating_projects for instructions on how to update projects.

CHAPTER 3

Upgrade Scenario Configuration

There is quite a bit more flexibility and capability in defining an upgrade scenario than was discussed in the *tutorial*. Here we will go through each field in the **Apply Upgrade** measure and discuss how it can be used to build more complicated real-life scenarios for upgrades.

3.1 Upgrade Name

This is a human readable name for the upgrade scenario. Something like, "Replace electric furnaces with Energy Star heat pumps" or "Insulate attics to R-49".

3.2 Option <#>

In this field we enter the parameter and option combination to be applied. In the upgrade scenario simulations, this option will replace the option for the corresponding parameter in the baseline run. These can be found and referenced in the `resources/options_lookup.tsv` file in your local git repository. (You can see the most updated version on github here, but it's recommended to use your local version as it will be synchronized with your project.) The file can be opened in a spreadsheet editor like Excel for viewing.

The text to enter in the field will be the Parameter Name followed by the Option Name separated by a pipe character.

```
Insulation Wall|Wood Stud, R-36
```

3.3 Option <#> Apply Logic

The apply logic field specifies the conditions under which the option will apply based on the baseline building's options. To specify the condition(s) include one or more `parameter|option` pairs from `options_lookup.tsv`. Multiple option conditions can be joined using the following logical operators. Parentheses may be used as necessary as well.

`||`	logical OR
`&&`	logical AND
!	logical NOT

A few examples will illustrate. First, lets say we want the apply the option `Water Heater|Gas Tankless`, but only for water heaters that are worse and also use gas. We would use the following apply logic:

```
Water Heater|Gas Standard||Water Heater|Gas Benchmark
```

Or say we want to apply the upgrade only to houses with 3 car garages that aren't in New England.

```
(!Location Census Division|New England)&&(Geometry Garage|3 Car)
```

Todo: Come up with some better examples here.

Currently, you can enter up to 25 options per upgrade. To allow additional options per upgrade you would need to update a method defined in a resource file, run a rake task, and update the outputs section for all PAT projects. See *Increasing Upgrade Options* for more information.

3.4 Option <#> Cost <#>

This is the cost of the upgrade. Multiple costs can be entered and each is multiplied by a cost multiplier, described below.

3.5 Option <#> Cost <#> Multiplier

The cost above is multiplied by this value, which is a function of the building. Since there can be multiple costs (currently 2), this permits both fixed and variable costs for upgrades that depend on the properties of the baseline house.

- Fixed (1)
- Conditioned Floor Area (ft^2)
- Conditioned Foundation Slab Area (ft^2)
- Lighting Floor Area (ft^2)
- Above-Grade Conditioned Wall Area (ft^2)
- Above-Grade Total Wall Area (ft^2)
- Below-Grade Conditioned Wall Area (ft^2)
- Below-Grade Total Wall Area (ft^2)
- Window Area (ft^2)
- Roof Area (ft^2)
- Door Area (ft^2)
- Water Heater Tank Size (gal)
- HVAC Cooling Capacity (kBtuh)

- HVAC Heating Capacity (kBtuh)

3.6 Package Apply Logic

This is where to specifiy logic to determine whether the whole package of upgrades is applied (all of the options together). It uses the same format as *Option <#> Apply Logic*.

Todo: An example of when this might be useful would be nice.