
Scout Documentation

Chioke Harris and Jared Langevin

Jul 24, 2026

CONTENTS

1	Documentation Contents	3
1.1	An Overview of Scout	3
1.2	Quick Start Guide	4
1.3	Installation Guide	7
1.4	Web Interface Tutorials	13
1.5	Local Execution Tutorials	27
1.6	Analysis Approach	80
1.7	Heating and Cooling Load Components	85
1.8	ECM Definition Reference	88
1.9	EIA Updates	119

Scout is a software program that estimates the impacts of various energy conservation measures (ECMs) in the U.S. residential and commercial building sectors. To learn more about what Scout can do, check out [An Overview of Scout](#).

If you're eager to run a full Scout analysis, the [Quick Start Guide](#) is the best place to begin.

For guidance on using the [Scout web interface](#), check out the [Web Interface Tutorials](#).

Scout is a project of the U.S. Department of Energy [Building Technologies Office \(BTO\)](#) and is freely available for public use.

DOCUMENTATION CONTENTS

1.1 An Overview of Scout

1.1.1 What is Scout and who should use it?

- Scout is a software program that estimates the national energy savings, avoided CO₂ emissions, and operating cost impact potential of various *energy conservation measures (ECMs)* in the U.S. residential and commercial building sectors.
- With Scout, users can explore both the savings impact and cost-effectiveness of ECMs under multiple adoption scenarios across the entire U.S. or a subset of climate zone(s).
- Scout can help organizations with large building energy efficiency programs or portfolios frame their high-level program benefits and costs. Researchers, building managers, architects/engineers, and the general public can also use Scout to determine where certain energy-saving technologies or approaches of interest fit into the larger U.S. market for building energy efficiency.
- Scout is under active development by the U.S. Department of Energy Building Technologies Office (BTO). The latest release of the software can be downloaded or [forked](#) from the [GitHub repository](#).

1.1.2 Results specify U.S. energy, CO₂, and cost savings

- Scout yields the following national-scale results for individual ECMs:
 - primary energy savings (in quadrillion Btus),
 - avoided CO₂ emissions (in million metric tons),
 - operational cost savings, and
 - per unit cost effectiveness, expressed by multiple financial metrics including [internal rate of return \(IRR\)](#), [simple payback](#), [cost of conserved energy \(CCE\)](#), and cost of conserved CO₂ (CCC) (conceptually equivalent to CCE, but should be compared with CO₂ prices).
- Results may be aggregated across multiple ECMs and filtered by variables such as climate zone, building type/vintage, and end use.

1.1.3 Scout ECMs and their baseline markets

- ECMs represent building technologies or operational approaches that improve upon the unit efficiency and/or lifetime operational costs of the comparable incumbent or “business-as-usual” technology or approach.
- *ECMs are characterized* by their energy efficiency, installed cost, lifetime, and applicable baseline markets; whether they add-on to or fully replace the service of a “business-as-usual” technology; whether they involve fuel switching; and their year of market entry and exit.

- The *applicable baseline market* for an ECM enumerates all of the *climate zones*, *building types*, *structure types*, *end uses*, *fuel types*, and *technologies* that are relevant to an ECM.
- ECM definitions can include distinct cost, efficiency, and lifetime values for each climate zone, building type, fuel type, end use, and/or technology type; probability distributions may also be placed on ECM cost, efficiency, and lifetime inputs.

See [EIA Updates](#) for details on how Scout aligns its baseline with the latest EIA Annual Energy Outlook.

1.1.4 ECMs are adopted one of two ways

There are two different scenarios that are used to represent consumer adoption of an ECM.

- The technical potential scenario assumes that as soon as an ECM is introduced, the entire baseline market instantaneously and completely switches to the new ECM, and the ECM retains a complete sales monopoly in subsequent years. Results from this scenario represent the maximum impact an ECM could have, limited only by baseline market size.
- The max adoption potential scenario assumes an ECM is only able to capture the portion of its baseline market associated with new construction and retrofit or replacement of existing equipment in a given year. Results from this scenario represent an ECM's maximum impact considering typical building and equipment turnover and generally show a gradual accumulation of ECM savings over time.

1.1.5 Competition of overlapping ECM markets

- In cases where multiple ECMs apply to the same baseline market, the market is apportioned among the competing ECMs based on each ECM's incremental capital costs and operating cost savings.
- In general, ECMs with lower incremental capital costs and higher operating cost savings will capture a greater share of a baseline market being competed. The importance of capital and operating cost savings on market share are weighted differently depending on the end use affected.
- ECM competition is needed to ensure there is no double-counting of energy, CO₂, and operating cost savings impacts when aggregating results across multiple ECMs.

1.2 Quick Start Guide

If you're new to Scout, this is the right place to get started. Steps for running a Scout analysis from start to finish are listed below; additional guidance on certain steps can be found in linked tutorials.

1.2.1 1. Setup Scout and prerequisites

Download the latest version of Scout and follow the [Installation Guide](#) to configure the programs required for Scout to run.

Note

If you'd like to execute the full set of standard Scout measures (ECMs) that are included with the installation in the folder `./ecm_definitions`, download the file `Latest_BM_Shapes.zip` [here](#) and unzip/add its contents to the folder `./ecm_definitions/energyplus_data/savings_shapes`.

1.2.2 2. Create new ECM definition(s) (optional)

Visit the Scout web interface's [ECM Summaries Page](#). Register and/or sign in to your Scout account using the “Register” or “Sign In” button at the top right of the page. Once signed in, click the “Add ECM” dropdown at the top right of the page and select the “Write New ECM” option. Follow the steps for generating a new ECM definition; the ECM definition file will be downloaded to your computer. Move the downloaded ECM to the `./ecm_definitions` folder (see Web Interface [Tutorial 2: Creating and editing ECMs](#) for additional guidance).

Alternatively, follow Local Execution [Tutorial 1: Creating and editing ECMs](#) to develop your own ECM definition(s) using a text editor.

1.2.3 3. Open the command line interface

Open a Terminal window (Mac)¹ or command prompt (Windows)² and navigate to the Scout project directory (shown with the example location `./Documents/projects/scout-0.1` – substitute the path to your Scout directory) by entering the following command line argument:

Windows

```
cd Documents\projects\scout-0.1
```

Mac

```
cd Documents/projects/scout-0.1
```

1.2.4 4. Run using project configuration file(s) (optional)

Scout scenarios can be defined by directly passing arguments to Python scripts, or with the use of a `.yaml` configuration file, which stores argument values in a consistent, trackable manner. This configuration file approach also provides the benefit of storing argument values for both core modules of Scout—`ecm_prep.py` and `run.py`—in one place. Guidance on creating configuration files is found in [Tutorial 2: Running with project configuration files](#). To run a single scenario, enter the following command line arguments, where `<my_config.yaml>` refers to a custom configuration file:

Windows

```
python scout\ecm_prep.py --yaml <my_config.yaml>
python scout\run.py --yaml <my_config.yaml>
```

Mac

```
python3 scout/ecm_prep.py --yaml <my_config.yaml>
python3 scout/run.py --yaml <my_config.yaml>
```

¹ To open Terminal, press `-space` on your keyboard, begin typing “terminal” in the search bar that opens, and select Terminal from the list of programs that appear.

² To launch the command prompt, press `Win+R` on your keyboard, type “cmd” in the search bar that opens, and press Enter.

Note

If running scenarios with a configuration file, then steps 5-7 in the Quick Start Guide are not needed.

Running multiple configuration files

Batches of configuration files can also be run with the use of the `run_batch.py` module. This automatically runs both `ecm_prep.py` and `run.py` for a set of configuration files in a common directory. To run files in a batch, enter the following command line argument, where `<config_directory>` refers to a directory containing a set of custom configuration files (see *Running with multiple configuration files* for further guidance):

Windows

```
python scout\run_batch.py --batch <config_directory>
```

Mac

```
python3 scout/run_batch.py --batch <config_directory>
```

1.2.5 5. Prepare ECMs for analysis

To prepare measures for *EIA Electricity Market Module (EMM)* regions, enter the following command line argument (see Local Execution *Tutorial 3: Preparing ECMs for analysis* for additional guidance and execution *options*):

Windows

```
python scout\ecm_prep.py
```

Mac

```
python3 scout/ecm_prep.py
```

Note

The standard set of ECM definitions included in `./ecm_definitions` requires the EMM region setting to execute. Only new or edited ECM definitions are updated in this step.

Tip

Preparing the full set of standard ECM definitions in `./ecm_definitions` will take several minutes. For a quicker test run, consider restricting the contents of this folder to just one or a handful of measures of interest while setting the contents of the file `./ecm_definitions/package_ecms.json` to a blank list `[]`.

1.2.6 6. Modify active list of ECMs (optional)

Enter the following command line argument (see Local Execution *Tutorial 4: Modifying the active ECMs list* for additional guidance):

Windows

```
python scout\run_setup.py
```

Mac

```
python3 scout/run_setup.py
```

1.2.7 7. Run analysis on active ECMs

Enter the following command line argument (see Local Execution *Tutorial 5: Running an analysis* for additional guidance and execution *options*):

Windows

```
python scout\run.py
```

Mac

```
python3 scout/run.py
```

1.2.8 8. View results plots and data

Open the `./results/plots` folder to view local plots of your results and access underlying data in Excel (see Local Execution *Tutorial 6: Viewing and understanding outputs* for additional guidance). Local plots are organized in folders by *adoption scenario* and *plotted metric of interest* (i.e., `./results/plots/(adoption scenario)/(metric of interest)`). Raw data for each adoption scenario's plots are stored in the XLSX files beginning with "Summary_Data."

1.3 Installation Guide

Before you can use Scout, you'll need to install a few things that Scout relies upon to run. Preparing for and using Scout requires interacting a bit with the command line, but these instructions will walk through each step in the set up process with the specific commands required. While the basic prerequisites are the same for *Mac* and *Windows* users, because the details and order of the steps are somewhat different, separate instructions are provided. Before beginning, you'll need to be using a computer where you have administrator-level privileges so that you can install new software.

1.3.1 Getting Scout

The first step is to clone Scout from GitHub. Choose the appropriate command based on your use case:

For using Scout (recommended for most users), clone the latest stable release.

On Mac/Linux:

```
git clone --branch $(git ls-remote --tags --sort=v:refname https://github.com/trynthink/scout.git | tail -n1 | sed 's/.*\///;s/\^{//'}) https://github.com/trynthink/scout.git
```

On Windows (PowerShell):

```
$tag = ((git ls-remote --tags --sort=v:refname https://github.com/trynthink/scout.git)[-1] -split '/')[-1] -replace '\^{\}', ''; git clone --branch $tag https://github.com/trynthink/scout.git
```

Or simply clone with the current latest tag (v1.2):

```
git clone --branch v1.2 https://github.com/trynthink/scout.git
```

For development, clone the master branch:

```
git clone https://github.com/trynthink/scout.git
```

After cloning, navigate to the Scout directory:

```
cd scout
```

If you're comfortable at the command line, install or set up everything in this list of prerequisites and then skip straight to [step 2](#) of the Quick Start Guide.

Prerequisites

- Python 3
- Scout Python package: `pip install .` from your Scout install directory
- A text editor of your choice

The installation instructions for [Mac](#) and [Windows](#) assume that none of these prerequisite programs or distributions are installed on your system. Please follow the instructions as appropriate, given what might already be installed on your system and checking for updates if appropriate.

Warning

Please use due care and take appropriate security precautions to ensure that your system is not compromised when installing the programs and distributions identified below. It is your responsibility to protect the integrity of your system. *Caveat utilitor.*

Mac OS

1.3.2 0. (Optional) Install a package manager

Installing and using a package manager will make it easy to ensure that multiple installations of Python do not interfere with each other. [Homebrew](#) is a popular package manager, but there are other options, such as MacPorts and Fink.

Note

While this step is optional, subsequent instructions are written with the assumption that you have installed Homebrew as your package manager.

To install Homebrew, open Terminal (found in Applications/Utilities, or trigger Spotlight with -space and type “Terminal”). Visit the [Homebrew website](#) and copy the installation command text on the page. Paste the text into the Terminal application window and press Return. If you encounter problems with the installation, return to the Homebrew website for help or search online for troubleshooting assistance.

If you are using a package manager other than Homebrew, follow the documentation for that package manager to install Python 3. If you have chosen to not install a package manager, you may use the [Python Software Foundation installer](#) for the latest version of Python 3.

1.3.3 1. Set Up a Python Virtual Environment

In a Terminal window, at the command prompt (a line terminated with a \$ character and a flashing cursor), install Python by typing:

```
brew install python3
```

A virtual environment should be used to ensure that package dependencies are isolated for Scout. Common tools for managing virtual environments include [venv](#), [virtualenv](#), and [conda](#). To create and activate a virtual environment named `.scout_env` using the standard Python module, `venv`, navigate to the Scout installation directory and run the following commands:

```
python3 -m venv .scout_env
source .scout_env/Scripts/activate
```

This will activate the new Python environment where the Scout package will be installed. To check that the environment was set up correctly, run `which python`, and ensure that the path points to the new environment directory.

Note

Each time you intend on running Scout, the virtual environment must be activated.

1.3.4 2. Install Scout Python package

Once Python 3 is fully installed and a virtual environment is activated, `pip3`¹ is the tool you will use to install add-ons specific to Python 3. Run the following from your Scout installation directory to install the Scout package:

```
pip3 install .
```

Note

For developers: if you intend on editing files within the Scout package directory, such as `scout/supporting_data` or `.py` files, run `pip3 install -e ".[dev]"` to install in editable mode with developer dependencies.

The Python packages Scout needs are listed under “dependencies” in the `pyproject.toml` file. If you’d like to confirm that the dependencies were installed successfully, you can run the command below to review the dependencies installed to your environment.

```
pip3 list
```

1.3.5 3. Install a text editor

A third-party text editor will make it easier to change Scout files. There are [many different text editors](#) available for the Mac. Mac OS X ships with two command line interface editors, `vim` and `emacs`. You may use one of these or install and use another graphical or command line interface editor of your choice. Whatever editor you choose should have support for syntax-specific code coloring and syntax-specific formatting and there should be [linting](#) for Python and JSON built-in or available through add-on packages. Python code linting should include checking for compliance with [PEP 8](#) (using the `pycodestyle` package) and `pyflakes`, at a minimum.

For the purposes of this documentation, the following instructions will step through how to install [Sublime Text](#), an easy to use text editor with a graphical interface that can be configured to satisfy the specified requirements. These instructions are provided to illustrate the steps required to configure a text editor for viewing and modifying Python and JSON files and should not be construed as an endorsement or promotion of Sublime Text.

¹ `pip/pip3` is typically installed at the same time that Python 3 is installed.

1. Download Sublime Text

To set up Sublime Text for working with Scout, [download Sublime Text 4](#), open the downloaded disk image, and drag the application file to the Applications folder using the shortcut provided.

After installing Sublime Text, there are several additional configuration steps that will help get the editor ready for viewing and editing Python and JSON files.

2. Install Package Control

First, open Sublime Text and, following the [directions](#) provided by the developer, install Package Control.

Once installed, Package Control is opened via the Command Palette (Tools > Command Palette or P). Begin typing “Package Control” into the Command Palette. If a list of options beginning with “Package Control” appear, then the installation was successful. If not, refer back to the [Package Control website](#) for troubleshooting help.

We will use Package Control to install the additional features needed for checking Python files.

3. Install SublimeLinter prerequisites

Before proceeding further, open a Terminal window and at the command prompt, use pip3 to install the pycodestyle and pyflakes packages:

```
pip3 install pycodestyle
pip3 install pyflakes
```

4. Install SublimeLinter

Return to Sublime Text and open Package Control using the Command Palette (Tools > Command Palette or P). Begin typing “Package Control: Install Package” in the Command Palette and click that option once it appears in the list. (Arrow keys can also be used to move up and down in the list.) In the search field that appears, begin typing “SublimeLinter” and click the package when it appears in the list to install the package. If installation was successful for this (or any other) package, the package name will appear in the Preferences > Package Settings sub-menu.

5. Install specific code linters

Open the Command Palette and select “Package Control: Install Package” again to install new packages following the same steps. Install the “SublimeLinter-pycodestyle,” “SublimeLinter-json,” and “SublimeLinter-pyflakes” packages.

6. Configure Python syntax-specific preferences

Finally, the Python-specific settings for Sublime Text need to be updated. Open a new file in Sublime Text and save it with the file name asdf.py. (asdf.py will be deleted later.) Open the Python syntax-specific settings (Sublime Text > Preferences > Settings – Syntax Specific) and between the braces, paste:

```
"spell_check": true,
"tab_size": 4,
"translate_tabs_to_spaces": true,
"rulers": [80]
```

Save the modified file and close the window. Once complete, delete asdf.py.

Quit and reopen Sublime Text to apply all of the settings changes and new packages that have been installed.

Windows

1.3.6 0. Determine whether you have 32-bit or 64-bit Windows installed

Some of the software prerequisites for Scout have different versions for 32-bit and 64-bit installations of Windows. If you are unsure of whether your computer is running 32-bit or 64-bit Windows, you can follow [these instructions](#) from Microsoft to find out.

1.3.7 1. Set Up a Python Virtual Environment

Tip

If you have 64-bit Windows installed on your computer, downloading and installing the 64-bit version of Python is recommended.

Download the executable installer for Windows available on the Python Software Foundation [downloads page](#). Run the installer and follow the on-screen prompts as you would with any other software installer. Be sure that the option in the installer “Add Python 3.x to PATH,” where x denotes the current version of Python 3, is checked.

A virtual environment should be used to ensure that package dependencies are isolated for Scout. Common tools for managing virtual environments include [venv](#), [virtualenv](#), and [conda](#). Begin by [opening a command prompt](#) window. To create and activate a virtual environment named `.scout_env` using the standard Python module, `venv`, navigate to the Scout installation directory and run the following commands:

```
py -3 -m venv .scout_env
.scout_env\Scripts\activate
```

This will activate the new Python environment where the Scout package will be installed. To check that the environment was set up correctly, run `where python`, and ensure that the first path returned points to the new environment directory.

Note

Each time you intend on running Scout, the virtual environment must be activated.

1.3.8 2. Install Scout Python package

Once Python 3 is installed and a virtual environment is activated, the Scout package and its dependencies can be installed. `pip`^{Page 9, 1} is the tool you will use to install add-ons specific to Python 3. Run the following from your Scout installation directory to install the Scout package:

```
python -m pip install .
```

Note

For developers: if you intend on editing files within the Scout package directory, such as `scout/supporting_data` or `.py` files, run `python -m pip install -e ".[dev]"` to install in editable mode with developer dependencies.

The Python packages Scout needs are listed under “dependencies” in the `pyproject.toml` file. If you’d like to confirm that the dependencies were installed successfully, you can run the command below to review the dependencies installed to your environment.

```
python -m pip list
```

1.3.9 3. Install a text editor

While Windows comes with a plain text editor, Notepad, there are [many different text editors](#) available for Windows that will make it much easier to view and change Scout files. You are welcome to use the editor of your choice, but whatever you choose should have support for syntax-specific code coloring and syntax-specific formatting and there should be [linting](#) for Python and JSON built-in or available through add-on packages. Python code linting should include checking for compliance with [PEP 8](#) (using the [pycodestyle](#) package) and [pyflakes](#), at a minimum.

[Sublime Text](#) is an easy to use cross-platform text editor that can be configured to have the necessary features for authoring Python and JSON files. The following instructions are provided to illustrate the steps required to configure a text editor for viewing and modifying Python and JSON files and should not be construed as an endorsement or promotion of Sublime Text.

1. Install Sublime Text

To set up Sublime Text for working with Scout, [download Sublime Text 4](#) and run the installer. The installer will automatically place the application and supporting files in the appropriate locations on your system.

After installing Sublime Text, there are several additional configuration steps that will help get the editor ready for viewing and editing Python and JSON files.

2. Install Package Control

First, open Sublime Text and, following the [directions](#) provided by the developer, install Package Control.

Once installed, Package Control is opened via the Command Palette (Tools > Command Palette or Ctrl+Shift+P). Begin typing “Package Control” into the Command Palette. If a list of options beginning with “Package Control” appear, then the installation was successful. If not, refer back to the [Package Control website](#) for troubleshooting help.

We will use Package Control to install the additional features needed for checking Python files.

3. Install SublimeLinter prerequisites

Before proceeding further, [open a command prompt](#) window and type the following commands to use pip to install the [pycodestyle](#) and [pyflakes](#) packages:

```
python -m pip install pycodestyle
python -m pip install pyflakes
```

Once you have

4. Install SublimeLinter

Return to Sublime Text and open Package Control using the Command Palette (Tools > Command Palette or Ctrl+Shift+P). Begin typing “Package Control: Install Package” in the Command Palette and click that option once it appears in the list. (Arrow keys can also be used to move up and down in the list.) In the search field that appears, begin typing “SublimeLinter” and click the package name when it appears in the list to install the package. If installation was successful for this (or any other) package, the package name will appear in Preferences > Package Settings.

5. Install specific code linters

Open the Command Palette and select “Package Control: Install Package” again to install new packages following the same steps. Install the “SublimeLinter-pycodestyle,” “SublimeLinter-json,” and “SublimeLinter-pyflakes” packages.

6. Configure Python syntax-specific preferences

Finally, the Python-specific settings for Sublime Text need to be updated. Open a new file in Sublime Text and save it with the file name `asdf.py`. (`asdf.py` will be deleted later.) Open the Python syntax-specific settings (Preferences > Settings – Syntax Specific) and between the braces, paste:

```
"spell_check": true,
"tab_size": 4,
"translate_tabs_to_spaces": true,
"rulers": [80]
```

Save the modified file and close the window, then delete `asdf.py`.

Quit and reopen Sublime Text to apply all of the settings changes and new packages that have been installed.

1.4 Web Interface Tutorials

1.4.1 Tutorial 1: Navigating the Landing Page

The [Landing Page](#) provides a single access point to all other pages in the Scout web interface and to additional information and resources on Scout.

Upon first visiting the Landing Page, you'll see a brief summary of what Scout does and an option to "Read More". Clicking the "Read More" button directs you to the [Welcome Page](#) of this documentation.

Scrolling down further, you'll see brief descriptions of each of the three other pages in the Scout web interface: [ECM Summaries](#), [Analysis Results](#), and [Baseline Energy Calculator](#). Clicking the "View ..." links underneath each description will direct you to the page being described.

At the bottom of the landing page, a section titled "Do more with Scout" points you to resources for running your own Scout analysis from start to finish ("Quick Start Guide" button), contributing to Scout's development on GitHub ("Contribute" button), and reporting any issues or questions you may have ("Report an Issue" button).

The latest version of the Scout source code is accessed by clicking the "Source Code" button at the top right of the Landing Page and all other pages in the web interface.

Tip

We recommend registering an e-mail address and creating an account so that you can save your ECMs and custom analyses. To create an account, click the "Register" button on the landing page and complete the required fields.

Note

To change your username or password, after you are signed in, hover over your username and click "Profile". On this page you will be able to change your username and/or password.

1.4.2 Tutorial 2: Creating and editing ECMs

The [ECM Summaries Page](#) allows you to create your own Scout ECM definitions; browse, filter and edit an existing set of ECM definitions; and upload your own set of ECMs.

Creating a new ECM or ECM package

On the ECM Summaries page, click the “Add ECM” dropdown towards the top of the page, and subsequently click “Write New ECM”. A web form will pop up that guides you step-by-step through the process of creating a new single ECM or ECM package definition. If you wish to create a single ECM definition, ensure that “Single ECM” is toggled at the top right of the form; if you wish to create an ECM package, toggle the “ECM Package” option at the top right of the form.

All inputs to the Add New ECM form are required unless they are tagged “optional”. Refer to red error messages below missing input(s) to ensure that your entries are complete.

Single ECM creation

The Add New ECM form guides you through seven steps for creating a single ECM definition, outlined below. Steps are navigated by either clicking the arrows at bottom left of the form or by clicking through the navigation bar on the left side of the form.

1. **Enter general information about the ECM.** Inputs include ECM name, a short description of the ECM, and an indication of whether the ECM is of the “Service Replacement” or “Add-On” type.

Note

A “Service Replacement” ECM fully replaces the service of a comparable “business-as-usual” technology (e.g., an efficient air source heat pump that replaces an existing gas furnace); an “Add-On” ECM enhances the service of the “business-as-usual” technology (e.g., a controls technology that improves the operational efficiency of an HVAC system). See [Add-on type ECMs](#) for additional guidance on this input.

Tip

Drag-and-drop an existing ECM JSON definition onto the gray box towards the bottom of the window to populate all remaining fields in the form.

2. **Define the ECM’s applicable baseline market.** An ECM’s applicable baseline market is defined by climate zone(s), building type(s), building vintage(s), end use(s), fuel type(s), and technology (technologies). Each of these baseline market attributes has multiple categories that are selected using checkboxes or radio buttons, including an “All” option that will automatically select all categories for the given market attribute.

Initially, only the “Climate Zone,” “Building Type,” and “Building Vintage” inputs are shown. Once you select a building type, the “End Use” input will appear; upon selecting most end uses¹, the “Fuel Type” input will appear; and upon selecting a fuel type, the “Technology” input will appear.

Tip

If the ECM impacts a whole end use or end uses (e.g., the ECM achieves 30% lighting end use savings), choose the appropriate “End Use” option(s) and subsequently set “Fuel Type” (if applicable) and “Technology” to “All”; this will ensure that ECM performance improvements assigned in step 4 are applied across the desired end use(s).

¹ In the special case of a “Heating,” “Secondary Heating,” and/or “Cooling” end use selection, an additional “Technology Type” input will appear, as heating/cooling technologies may be categorized as equipment (e.g., an efficient air source heat pump) or as envelope components (e.g., a highly insulating window). Choosing “Equipment” as the “Technology Type” will yield subsequent “Fuel Type” and “Technology” inputs. Choosing “Envelope” as the “Technology Type” skips the “Fuel Type” input and moves you straight to the “Technology” input; this reflects that the energy use associated with envelope components is not tied to a specific fuel type.

3. **Enter the ECM's market entry and exit year.** *Optionally*, enter the years that the ECM is expected to enter and exit the market in the "Market Entry Year" and "Market Exit Year" boxes.

Note

If you choose not to enter a market entry and exit year for the ECM, the ECM's market entry year is assumed to be the first year of the modeling time horizon and no market exit year is assumed.

Enter supporting source information in the "Market Entry Year Source" and "Market Exit Year Source" boxes. In each source box, first enter general notes about the source, followed by one or more associated URLs. Source notes and URLs should be separated by semicolons, for example:

"Sample source title, author/organization, and year. Additional notes about what is included in the source; sampleurl1.com; sampleurl2.com"

4. **Enter the ECM's energy performance level.** Energy performance level data may be entered using either a simple method (default) or a detailed method that accommodates different performance value types and breakdowns.

Simple entry of energy performance data

- If you wish to use *absolute units* for energy performance (e.g., COP), enter the absolute performance value in the box just below the "Energy Performance" label.

Note

Absolute performance values must reflect the units autofilled in the gray box to the right of the value box. Performance units are auto-filled based on your selections in step 2 - see [Energy efficiency units](#) for additional guidance.

- If you wish to use *relative savings* units for energy performance (e.g., % savings), use the drop down menu to choose between "constant" and "dynamic" relative savings options and enter the percent savings value in the box just below the "Energy Performance" label. When selecting "dynamic" relative savings units from the drop down menu, enter the anchor year into the box that appears to the right of the units drop down menu.

Note

"Constant" relative savings units indicate that the ECM's relative savings impact does not change over time, even as the baseline technology stock becomes more efficient. Conversely, "dynamic" units recalculate the ECM's relative savings impact annually to reflect a changing baseline, using an anchor year to determine the magnitude of baseline change over time. See [Relative energy efficiency units](#) for additional guidance.

Detailed entry of energy performance data

- Toggle the "Detailed Input" option shown towards the top right of the form. Click the "Add Data" button to begin specifying the detailed inputs. A second form will appear that allows you to breakdown energy performance values by energy use segment ("Breakdown by Segment"), change the type of energy performance value ("Change Value Type"), and set the energy performance value and associated units ("Set Value" and "Set Units"). Performance units are specified in either absolute terms - with units autofilled from step 2 - or as a relative savings percentage.

Note

In the “Add Data” form, you can adjust default selections for “Breakdown by Segment” without adjusting the default selection for “Change Value Type” in this form, and vice versa.

- Energy performance values are specified as point values by default (“Point” option under “Change Value Type”). Alternatively, a [probability distribution](#) may be placed on the performance values (“Probability Distributions” under “Change Value Type”). When a probability distribution is placed on energy performance, the “Set Value” box will prompt you for the chosen distribution’s parameters.

Note

Users may notice the “EnergyPlus” option when exploring the “Change Value Type” drop down. This option will eventually enable users to load an ECM’s performance input value from an external [EnergyPlus Measure](#) simulation. However, this feature is currently unsupported in the Scout analysis engine and users are therefore discouraged from selecting the “EnergyPlus” option in the “Change Value Type” drop down.

- Additional rows of information may be added to the window using the “Add New Fields” button at bottom left; existing rows may be deleted using the trash bin icon to the right of each row.

Tip

When making selections in the “Breakdown by Segment” drop down menus, ensure that a complete set of breakdowns is entered. For example, if an energy performance level is broken down by the “New” building vintage type in one row, a second performance level must be specified for the “Existing” building vintage type in another row.

- When you are done making your detailed selections, click the “Submit Data” button at the bottom right of the form; your selections are now summarized in a table under the “Energy Performance” label that includes “Value Breakdown,” “Value Type,” “Value,” and “Units” columns.

Enter energy performance source information in the “Energy Performance Source” box using the same source content and formatting guidelines as in step 3.

5. **Enter the ECM’s installed cost.** Like energy performance data in step 4, installed cost data may be entered using the simple or detailed method.

Simple entry of installed cost data

- Enter the installed cost value in the box just below the “Installed Cost” label.

Note

Installed cost values must reflect the units autofilled in the gray box to the right of the value box. Cost units are auto-filled based on your selections in step 2 - see [Installed cost units](#) for additional guidance.

Detailed entry of installed cost data

- Toggle the “Detailed Input” option shown towards the top right of the form. Detailed input entry for installed cost works similarly to the process described for detailed energy performance inputs in step

4.

- Installed cost inputs may be broken out by building type and vintage, and may be specified as point values or probability distributions.

Enter installed cost source information in the “Installed Cost Source” box using the same source content and formatting guidelines as in step 3.

6. **Enter the ECM’s lifetime.** Like energy performance data in step 4 and installed cost data in step 5, lifetime data may be entered using the simple or detailed method.

Simple entry of lifetime data

- Enter the lifetime value in the box just below the “Lifetime” label. Lifetime is always specified in units of years.

Detailed entry of lifetime data

- Toggle the “Detailed Input” option shown towards the top right of the form. Detailed input entry for lifetime works similarly to the process described for detailed energy performance inputs in step 4.
- Lifetime inputs may be broken out by building type, and may be specified as point values or probability distributions.

Enter lifetime source information in the “Lifetime Source” box, using the same source content and formatting guidelines as in step 3.

7. **Enter other ECM information.** You may optionally specify inputs that:

- scale down an ECM’s applicable baseline market (“Market Scaling Fraction”),
- flag a switch between baseline technology fuel type(s) and the ECM’s fuel type (“Fuel Switching”),
- provide additional notes about the ECM definition (“Notes”), and
- identify the ECM definition’s author (“Author”).

Toggle the “Detailed Input” option shown to the top right of the “Market Scaling Fraction” input to access a detailed entry method for this input.

Note

When a market scaling fraction is specified, supporting source information for this fraction is required - see step 3 for source content and formatting guidelines.

Additional guidance is available for each of the inputs in step 7 in [Other fields](#).

Once all steps of the Add ECM form have been completed, click the “Generate ECM” button at bottom right to generate a single ECM JSON file and download the JSON to your computer; this JSON can be added to the `.ecm_definitions` folder in your Scout directory and used in subsequent analyses.

ECM package creation

For ECM Packages, the Add ECM form involves one step of data entry.

First, choose the single ECM definitions you wish to package from the list of options under “Select ECMs to Package”.

Tip

The “Select ECMs” list is populated from the default set of ECMs shown on the ECM Summaries Page; if you have a custom ECM stored in your local `./ecm_definitions` folder that you wish to package, add the ECM to the list by typing its name into the “Write in ECM” box and clicking the “Add” button at the right end of this box.

Enter a name for the ECM package in the “Package Name” box and add a brief description of the package in the “Package Description” box.

Next, enter any installed cost reductions or energy performance improvements yielded by packaging the single ECMs together in the “Installed Cost Reduction” and “Energy Performance” boxes. If no additional cost or performance benefits are expected from packaging, set these percentage values to zero.

Finally, enter supporting source information for the ECM package and its cost or performance benefits under the “Source” label, using the same source content and formatting guidelines as in step 3 of [Single ECM creation](#).

Once all the required inputs have been entered, click the “Generate ECM” button at bottom right to download a JSON definition of the ECM package. The ECM package definition will be added to an existing JSON file `package_ecms.json` that includes a default set of BTO ECM packages, and this file will be downloaded to your computer; again, this JSON can be added to the `./ecm_definitions` folder in your Scout directory and used in subsequent analyses.

Browsing and editing a set of ECMs

ECM definitions are presented in a table where each row includes information for a single ECM or ECM package and the columns summarize the following ECM attributes:

- name
- energy performance,
- installed cost,
- lifetime, and
- market entry year.

Filter and reorganize the ECM set

- **Search by ECM name.** Type part or all of the ECM name(s) of interest into the search bar at the top of the page and click the magnifying glass icon or press “Enter”.
- **Sort ECMs.** Click the arrow icons next to the “Name,” “Lifetime,” or “Market Entry Year” column headings to sort ECMs by each attribute in ascending order; click the arrow icons again to sort ECMs by each attribute in descending order.
- **Filter ECMs by end use, region, or building type.** Check the appropriate box(es) under the “End Use,” “Regions,” and/or “Building Class” headings in the left-hand filter bar; then click the “Apply Filter” button at the bottom of the filter bar.
- **Filter ECMs by total affected market.** Move each scroll bar dot under the “Total Affected Market” heading in the left-hand filter bar to set thresholds for the total energy use, CO₂ emissions, and/or energy costs affected by each ECM; then click the “Apply Filter” button at the bottom of the filter bar. Alternatively, enter a specific threshold or thresholds into the box to the right of each scroll bar and click “Apply Filter”.

Tip

Previous ECM filter selections are cleared by clicking the “Clear All” text at the bottom of the left-hand filter bar; the filter bar may also be hidden by clicking the left-pointing arrow shown at the top right of the bar.

View additional ECM details

To view additional details about an ECM on the ECM Summaries Page, click the drop down arrow to the left of each ECM's name. A new window will drop down that provides more data on ECM attributes and displays a series of line plots. Fig. 1 shows an example drop down window for a prospective Automated Fault Detection and Diagnosis (AFDD) ECM.

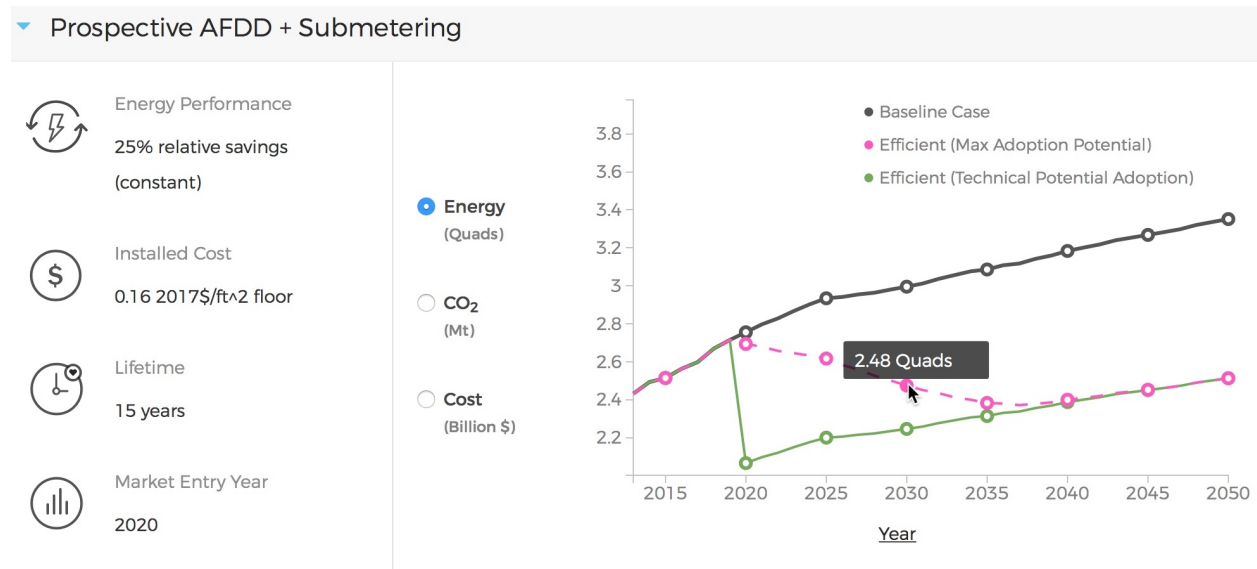


Fig. 1: Detailed data for a prospective AFDD ECM include key input attributes (at left) and primary energy use, CO₂ emissions, or energy cost results for the ECM under three ECM adoption scenarios (at right). In the energy use plot shown for this ECM, baseline primary energy use gradually increases across the modeled time horizon from 2.5 quads in 2015 to 3.35 quads in 2050; the ECM is ultimately able to reduce this baseline energy by more than 0.75 quads. The full impact of the ECM on baseline energy use is seen upon market entry in 2020 under a Technical Potential adoption scenario and by about 2040 under a Maximum Adoption Potential scenario.

Plots in the detailed ECM view show projected primary energy use, CO₂ emissions, or energy costs for the ECM under three ECM adoption scenarios (see *ECMs are adopted one of two ways* for more details on ECM adoption):

1. a “Baseline” technology case where no ECM adoption is assumed, corresponding to [AEO Reference Case](#) outcomes,
2. a “Technical Potential” case where the ECM is assumed to entirely replace comparable baseline technologies upon market entry, and
3. a “Maximum Adoption Potential” case where the ECM’s penetration into its baseline market is limited by more realistic baseline technology stock turnover.

Each plot’s x axis shows the year range for the projections; the y axis can be toggled to show the energy use, CO₂, or cost outcome of interest.

Tip

To view y axis values associated with each point on the plot, mouse over the points of interest and the values will appear.

Note

Plotted results on the ECM Summaries Page are estimated for each ECM in isolation - e.g., no competition with other ECMs is accounted for.

Download or edit ECM definitions

To download an ECM definition on the ECM Summaries Page, click the “Download” icon (the down arrow) at the right end of the row for the ECM of interest — this icon is also found at the top right of the detailed drop down window for the ECM. An ECM JSON will be downloaded to your computer; this JSON can be added to the `.ecm_definitions` folder in your Scout directory and used in subsequent analyses.

To edit the attributes of an ECM on the ECM Summaries Page, click the “Edit” icon (the pencil) at the right right end of the row for the ECM of interest — this icon is also found at the top right of the detailed drop down window for the ECM.

An “Edit ECM” form will pop up with fully populated input fields (see [Creating a new ECM or ECM package](#) for additional guidance on these fields). For edits to single ECMs, click through the navigation bar steps on the left side of the form and make changes to the input fields shown in each step; ECM package edits only have one step. When your edits are complete, click the “Generate ECM” button at the bottom right of the screen to download an edited ECM JSON definition; again, this JSON can be added to the `.ecm_definitions` folder in your Scout directory and used in subsequent analyses.

1.4.3 Tutorial 3: Creating new custom analyses

The [ECM Summaries Page](#) also allows you to create new, custom analyses with existing or new ECMs.

Create a new analysis with one or more ECMs from the ECM Summaries page

To conduct a new analysis, first select one or more of the ECM definitions on the [ECM Summaries Page](#) by clicking on the checkboxes next to the ECMs, and then enter a name for your analysis in the “Enter Analysis Name” box at the bottom of the screen.

Before you click “Start New Analysis,” which will no longer be greyed out after you enter a name for your analysis, you will need to pick an analysis calculation method. There are three calculation methods you can choose from depending on whether you are interested in [assessing source or site energy impacts](#):

1. **Fossil Fuel Equivalence:** One of two approaches for non-combustible source energy accounting, this methodology uses the average heat rate of fossil generators and assigns it as the heat rate for non-combustible renewable energy generation. This value is 9,510 BTU/kWh, or about 35% efficiency, and represents the source energy value of fossil generation that is displaced by renewable energy generation.
2. **Captured Energy:** The other approach assumes that the source energy of renewable energy generators is exactly equal to the electricity produced with no energy losses prior to transmission and distribution. It is equal to a heat rate of 3,412 BTU/kWh, or a conversion efficiency of 100%.
3. **Site Energy:** This approach assess impacts in terms of site energy rather than in terms of source energy (using one of the other two approaches).

Note

Neither of the source energy accounting methods is strictly more accurate than the other. Both are a matter of methodological choice related to the specific application. We recommend reviewing this [methodological guidance](#) for more information before making your choice.

After selecting one of these calculation methods, click “Start New Analysis,” and a pop-up box will indicate that the analysis has started and is running in the background. To view the analysis queue, hover your mouse over the queue icon where you will see an “Analysis Status” dropdown. Click on this dropdown to open an “Analysis Status” pane that shows your analysis is underway. When it is complete, you will see a new notification appear. The “Analysis Status” pane will now show your analysis has status “Completed”.

Using the analysis selection dropdown on the Analysis Results page

On the [Analysis Results Page](#) you can click the dropdown menu to show all completed and in-progress analyses. The pane will show the status, name, calculation method, and run date/time for each analysis. Next to each analysis, you will also see icons that allow you perform operations on the completed analyses:

- Clicking the green gear icon shows you the list of ECMs included in your analysis. If you wish to run a new analysis using all or a subset of these ECMs, you can use the checkboxes to select or deselect ECMs and then click “Next” to start a new analysis with a modified set of ECMs or with a different calculation method.
- Clicking on the blue download icon allows you to download in [JSON format](#) raw results from your analysis.
- Clicking on the blue edit icon allows you to rename the analysis.
- Clicking on the red delete icon allows you to delete the analysis.

1.4.4 Tutorial 4: Viewing and understanding outputs

The [Analysis Results Page](#) allows you to view interactive results visualizations for ECM analyses.

Energy, CO₂, cost, and financial metrics tabs

Results are organized by one of four outcome variables of interest - “Energy,” “CO₂,” “Costs,” and “Financial Metrics” - each of which is assigned a tab at the top of the page. Here, “Energy” signifies total primary energy use; “CO₂” and “Costs” signify the total CO₂ emissions and operating costs associated with the total primary energy use; and “Financial Metrics” cover various consumer and portfolio-level metrics of ECM cost effectiveness.

On each of the four tabs, summary statistics for the variable of interest are shown in a single line at the top of the page. All statistics correspond to a particular year range, which is shown in parentheses next to the title above the statistics. The year range can be changed using the “Year Range” boxes on the left-hand filter bar.

Tip

Entering an identical minimum and maximum year (e.g., 2020 to 2020) into “Year Range” yields annual results; otherwise, the “Energy,” “CO₂,” and “Costs” results are added across the specified year range, and the “Financial Metrics” results are averaged across the specified year range.

Summary statistics are tailored to each tab - for example, the statistics on the “Energy” tab begin with avoided energy use, while the statistics on the “CO₂” tab begin with avoided CO₂ emissions.

Each of the “Energy,” “CO₂,” and “Costs” tabs features two types of plots - a *radar graph* and a *bar graph*. Switch between these graph types using the “Radar” and “Bar” graph toggle towards the top right of these pages. The “Financial Metrics” tab features a single *scatterplot* type. Each of these visualizations is detailed further below.

Radar graphs

The [radar graph](#) on the Analysis Results Page groups total energy, CO₂, or cost results by end use, climate zone, or building class. The graph has several axes that emanate from a single point of origin; each axis represents a category for one of the three grouping variables. The magnitude of total energy, CO₂, or cost attributed to each category is represented by the distance of a point on the category’s axis from the axis origin. [Fig. 2](#) shows an example radar graph for a portfolio of prospective and commercially available ECMs.

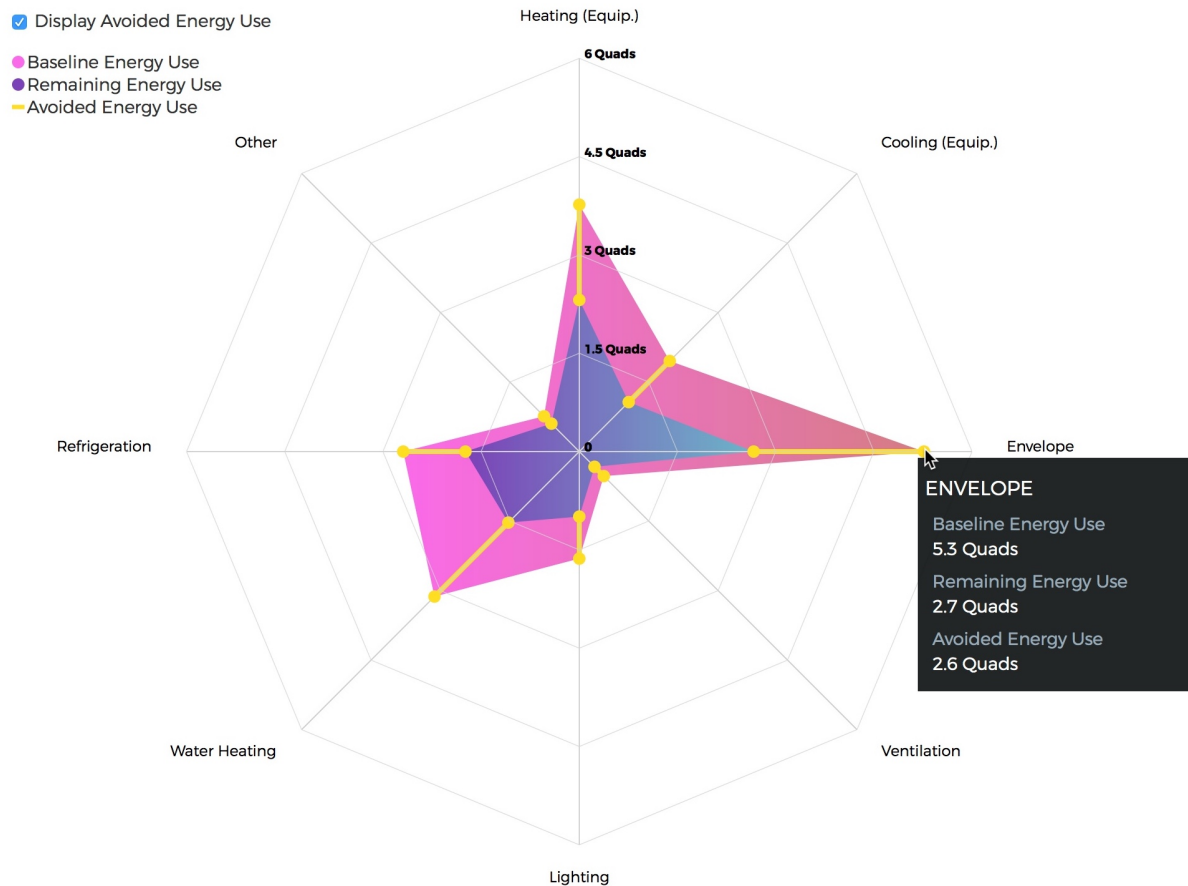


Fig. 2: In this radar graph, the overall primary energy use impact of an ECM portfolio that includes both prospective and commercially available ECMs is broken down by end use; results are shown for a Technical Potential adoption scenario run in the year 2030. Here, the envelope end use (pertaining to heating and cooling energy lost through building envelope components) makes the largest contribution to baseline energy use (5.3 quads) and ECM energy savings (2.6 quads).^{Page 23, 2} Heating and water heating yield the second and third largest baseline energy use totals (3.8 and 3.1 quads, respectively); ECM energy savings are slightly higher for water heating than heating (1.6 quads water heating compared to 1.5 quads heating).

Results are shown for two energy use scenarios:

1. a “business-as-usual” case where no ECM adoption is assumed, corresponding to [AEO Reference Case](#) outcomes (titled “Baseline” on the plot and shaded pink), and
2. a case where at least some degree of ECM penetration into the baseline energy, CO₂, or cost market is assumed (titled “Remaining” on the plot and shaded purple).

Note

Comparing the pink and purple shaded polygons on each radar graph - corresponding to the “Baseline” and “Remaining” scenarios, respectively - yields a visual understanding of which categories make the greatest contributions to avoided energy use, CO₂, or cost.

Tip

To view more details about the avoided energy use, CO₂, or cost attributed to a particular category, mouse over the yellow points and connecting line shown along the category’s axis; a tooltip will appear that summarizes baseline, remaining, and avoided energy use, CO₂, or cost.

The yellow points and lines may be removed from the plot by unchecking the “Display Avoided Energy Use” box towards the top left of the plot region.

In scenario 2 (“Remaining”) where some degree of ECM market penetration is assumed, ECM adoption is simulated one of two ways:

1. “Technical Potential” adoption, where the ECM entirely replaces comparable baseline technologies upon market entry, or
2. “Maximum Adoption Potential”, where the ECM’s penetration into its baseline market is limited by realistic baseline technology stock turnover (see [ECMs are adopted one of two ways](#) for more details on ECM adoption).

Switch between the end use, climate zone, and building class grouping variables by selecting the appropriate radio button under the “Group By” label on the left-hand filter bar and clicking the “Apply Filter” button at the bottom of the filter bar.

Change ECM adoption assumptions by adjusting the radio button under the “Adoption Scenario” label in the left-hand filter bar and clicking “Apply Filter” at the bottom of the filter bar.

Bar graphs

The [bar graph](#) on the Analysis Results Page attributes total avoided energy, CO₂, or cost results to individual ECMs. The plot unfolds from left to right, beginning with a “Baseline” segment of energy use, CO₂, or cost that is broken into “Avoided” and “Remaining” segments; each of these segments is then further attributed to the individual ECMs. [Fig. 3](#) shows an example bar graph for a portfolio of prospective and commercially available ECMs.

Tip

Clicking on the “Avoided” and “Remaining” segment bars shows the individual ECM contributions to each of these segments.

The magnitude of each ECM’s contribution to the “Avoided” and “Remaining” bar segments is indicated in three ways:

² EnergyPlus Measure performance data are always provided with units of relative savings.

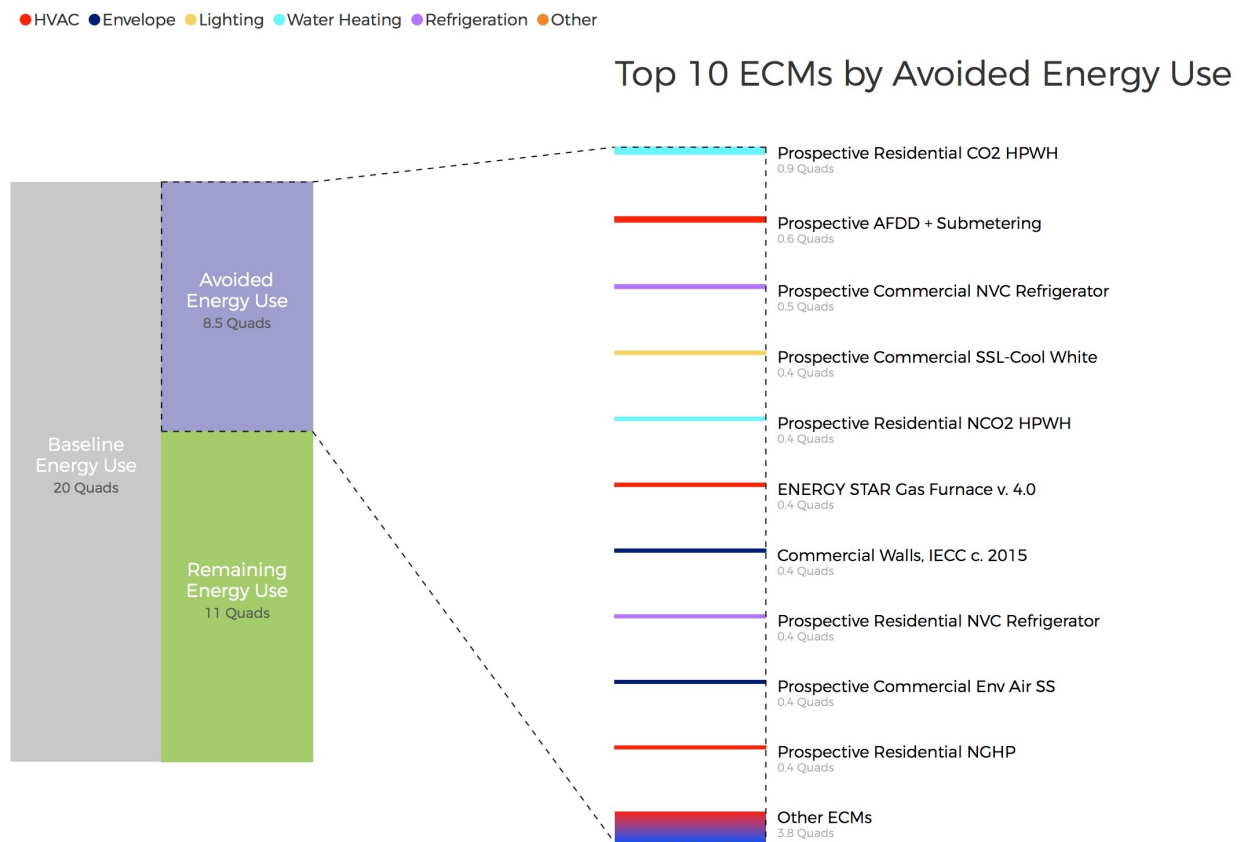


Fig. 3: This bar graph attributes total avoided and remaining energy use after ECM portfolio adoption to individual ECMs in the portfolio, grouping each ECM by the end use that it applies to. In this case, representing a Technical Potential ECM adoption run for the year 2030, two heat pump water heating (HPWH) ECMs appear in the top 5 ECM contributions to total avoided energy use. This result stems from the generally high potential for energy use impacts in the water heating end use (see Fig. 2) and the high performance levels of these HPWH ECMs relative to the baseline technologies they replace (here assumed to include gas-fired water heaters). Most of the ECMs on this list represent aspirational technologies with targeted cost and performance attributes.

1. by the ECM's vertical position on the list of individual ECMs, with more impactful ECMs shown higher on the list, and
2. by the height of each ECM's corresponding bar segment, and
3. by the specific energy, CO₂, or cost impact noted under the ECM's name label.

ECM bar segments may be color-coded by end use, climate zone, and building class; select the appropriate radio button under the "Group By" label on the left-hand filter bar and click the "Apply Filter" button at the bottom of the filter bar.

ECM bar segments may also be filtered by end use, climate zone, and building class by checking the appropriate box(es) under the "End Uses," "Climate Zones," and/or "Building Class" labels on the left-hand filter bar and clicking the "Apply Filter" button at the bottom of the filter bar.

As for the [ECM Summaries Page](#), previous ECM filter selections are cleared by clicking the "Clear All" text at the bottom of the filter bar; the filter bar may also be hidden by clicking the left-pointing arrow shown at the top right of the bar.

Scatterplots

The [scatterplot](#) on the Analysis Results Page indicates the cost effectiveness of individual ECMs under multiple financial metrics - Internal Rate of Return (IRR), Simple Payback, Cost of Conserved Energy, and Cost of Conserved Carbon - each of which may be assigned to the x or y axis of the plot using adjacent dropdown menus. [Fig. 4](#) shows an example scatterplot for a portfolio of prospective and commercially available ECMs.

Note

For all of the financial metrics except for IRR, a higher number signifies lower ECM cost effectiveness.

ECM points on the scatterplot may be color-coded by end use, climate zone, and building class; select the appropriate radio button under the "Group By" label on the left-hand filter bar and click the "Apply Filter" button at the bottom of the filter bar.

ECM points may also be filtered by end use, climate zone, and building class by checking the appropriate box(es) under the "End Uses," "Climate Zones," and/or "Building Class" labels on the left-hand filter bar and clicking the "Apply Filter" button at the bottom of the filter bar.

Tip

To see more details about an ECM's avoided energy use and CO₂ emissions, as well as the financial metric outcomes for an ECM under the current axis selections, hover your cursor over the ECM point of interest; a tooltip will appear with these details.

If a probability distribution has been placed on the cost, performance, and/or lifetime input(s) of one or more ECMs, uncertainty in the resultant cost effectiveness of any affected ECMs is expressed as a lightly shaded region around the affected ECM points on the scatterplot. Uncertainty ranges are also reported around the financial metric results in the detail tooltips of affected ECMs.

1.4.5 Tutorial 5: Using the Baseline Energy Calculator

The [Baseline Energy Calculator](#) allows you to determine the total energy and CO₂ impact potential of an individual ECM or group of ECMs, drawing data from the Energy Information Administration's [Annual Energy Outlook \(AEO\)](#).

The Calculator guides you through four steps to determining the total baseline energy use or CO₂ emissions associated with a particular market segment, shown below.

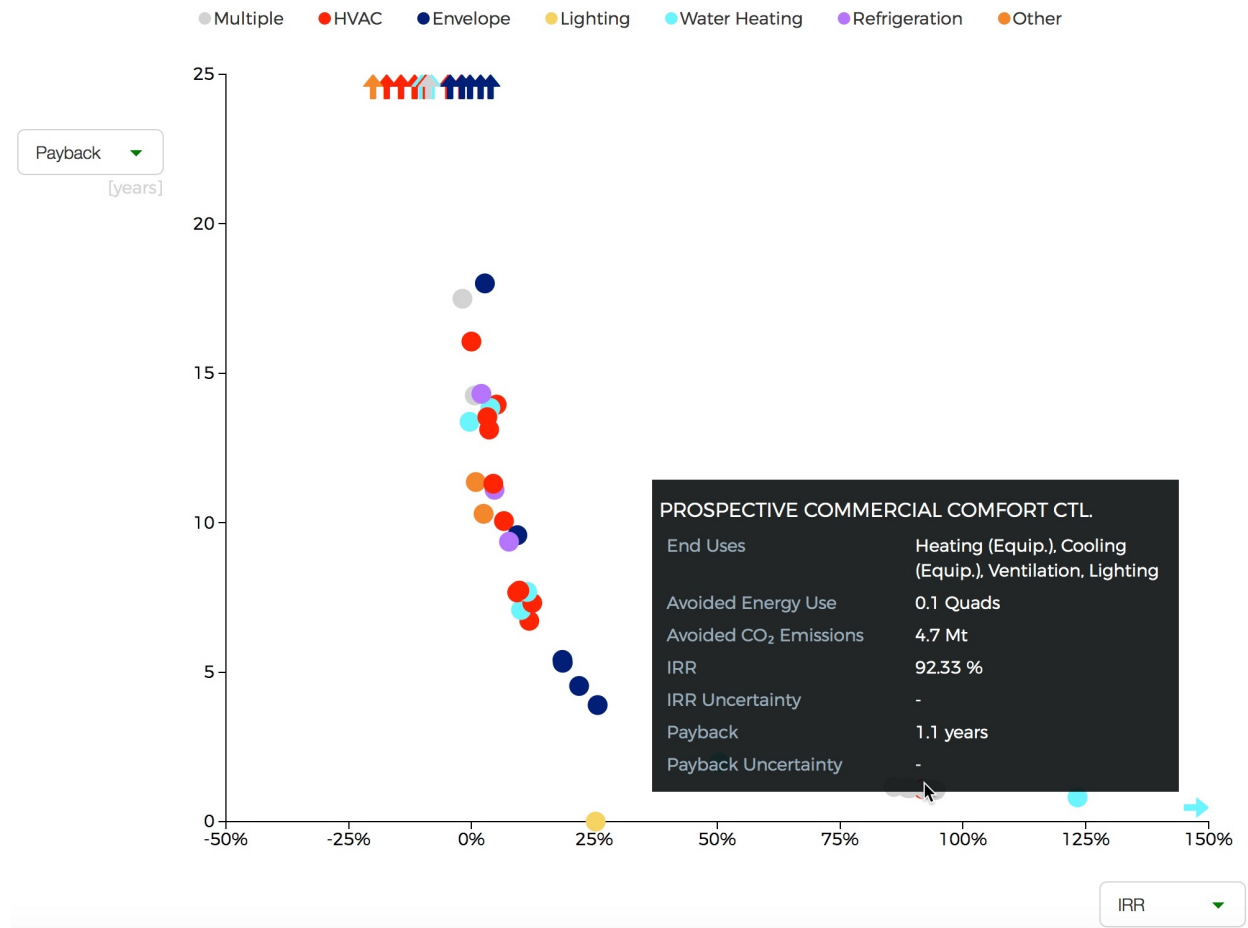


Fig. 4: This scatterplot indicates the cost effectiveness of individual ECMs under two financial metrics, grouping ECMs by the end use(s) they apply to. In this case, representing a Technical Potential ECM adoption run for the year 2030, internal rate of return (IRR) and simple payback financial metrics are used on the x and y axes, respectively. ECMs toward the bottom right of the plot (lower payback, higher IRR) are most cost effective. Most ECMs in the plot region below 5 years payback and above an IRR of 10% apply to the envelope, water heating, or “Multiple” end use categories, where the latter category reflects controls ECMs. Controls ECMs look particularly favorable here since their targeted cost and performance attributes were developed under a more aggressive payback requirement than other ECM types (~1 year). For example, the highlighted “Commercial Comfort Ctl.” ECM yields a 1.1 year payback in 2030, though this ECM only saves 0.1 quads of energy because its application was restricted to large offices for this run.

All inputs to the Baseline Energy Calculator form are required unless they are tagged “optional”. Refer to red error messages below missing input(s) to ensure that your entries are complete.

1. **Select an AEO version year.** Click the dropdown menu next to “AEO Year” to select which version of the EIA AEO you would like to use in generating results.
2. **Select a projection year.** Click the dropdown menu to select the year for which baseline energy or CO₂ emissions estimates are desired. Note: past years represent historical EIA estimates.
3. **Select relevant climate zone(s).** Climate zone(s) are selected by checking the appropriate box(es) or by clicking the region(s) of interest on the map. An “All” selection will automatically check all of the climate zone sub-categories.

Note

Scout currently uses the AIA climate zone breakdowns from [RECS 2009](#) and [CBECS 2003](#).

4. **Select building type(s).** Building type(s) are selected by checking the appropriate box(es). An “All Residential” or “All Commercial” selection will automatically check all residential and commercial building sub-categories, respectively.

Tip

Both residential and commercial building types may be selected simultaneously if you are interested in understanding impact potential across the entire buildings sector.³

5. **Select end use(s) and technology type(s).** End use(s) are selected by clicking the drop down menu bar and checking the appropriate box(es); click the drop down bar again to hide your end use selections and move on to subsequent selections. After most end use selections, a “Fuel Type” input will appear; upon selecting fuel type(s), a final “Technology” input will appear.⁴

Once all steps of the Baseline Energy Calculator have been completed, click the “Calculate” button at the bottom right of the screen to obtain the energy use and associated CO₂ emissions results.

Tip

Initial results may be cleared by clicking the “Reset” button or updated by clicking the “Calculate” button again.

1.5 Local Execution Tutorials

1.5.1 Tutorial 1: Creating and editing ECMs

Before beginning this tutorial, it is recommended that you review the [list of parameters](#) in an ECM definition.

The [ECM JSON schema](#) should always be in your back pocket as a reference. This section includes brief descriptions, allowable options, and examples for each of the fields in an ECM definition. You might want to have it open in a separate tab in your browser while you complete this tutorial, and any time you’re authoring or editing an ECM.

³ Envelope energy use totals in [Fig. 2](#) reflect simultaneous improvements in the efficiency of the heating and cooling equipment and lighting end uses (contributing 3.8, 2.0, and 1.6 quads of baseline energy use, respectively).

⁴ When both residential and commercial buildings are selected in step 3, subsequent end use and technology selections in step 4 will reflect both of these sectors. For example, in the end use dropdown menu, both “Secondary Heating” (residential only) and “Ventilation” (commercial only) end uses will be shown, while in the technology dropdown menu, “Wood Stove” (residential only) and “Engine-driven Heat Pump” (commercial only) will be shown.

The example in this tutorial will demonstrate how to write new ECMs so that they will be compliant with all relevant formatting requirements and provide the *needed information* about a technology in the structure expected by the Scout analysis engine. This example is intentionally plain to illustrate the basic features of an ECM definition and is *not* an exhaustive description of all of the detailed options available to specify an ECM. These additional options are presented in the *Additional ECM features* section, and the *ECM JSON schema* has detailed specifications for each field in an ECM definition. In addition, this tutorial includes information about how to *edit existing ECMs* and *define package ECMs*.

As a starting point for writing new ECMs, an empty ECM definition file is available for [download](#). Reference versions of the tutorial ECMs are also provided for download to check one's own work following completion of the examples.

Each new ECM created should be saved in a separate file. To add new or edited ECMs to the analysis, the files should be placed in the `.ecm_definitions` directory, or in a *directory specified by the user*. Further details regarding where ECM definitions should be saved and how to ensure that they are included in new analyses are included in [Tutorial 3](#).

JSON syntax basics

Each ECM for Scout is stored in a **JSON**-formatted file. JSON files are human-readable (they appear as plain text) and are structured as (nested) pairs of *keys* and *values*, separated by a colon and enclosed with braces.¹

```
{"key": "value"}
```

Warning

JSON files have important [formatting rules](#) that must be strictly observed. JSON files that do not follow these rules are “invalid” and cannot be used by any program.

Keys are *always* text – both letters and numbers are acceptable – enclosed in double quotes. Values can take one of several forms, such as numbers, boolean values (i.e., true, false), or lists (multiple values separated by commas and enclosed in brackets). A value must always be provided for each key.

```
{"temperature": 450}
{"activated": true}
{"preferred colors": ["red", "green", "blue"]}
{"reported heights": [1.79, 1.83, 1.64, 1.91]}
```

There can be multiple key-value pairs at the same level, separated by commas.

```
{"name": "HAL", "model": 9000, "service date": "1992-01-12"}
```

Values can also be additional key-value pairs, thus creating a nested structure.

```
{"vehicle 1": {"make": "Ford", "model": "F-150"}}
{"name": "HAL", "model": 9000, "service date": "1992-01-12",
 "manufacturing location": {"country": "US", "state": "IL", "city": "Urbana"}}
```

Among key-value pairs at the same level, the order of entries does not matter.

We will use these formatting guidelines to write new ECMs.

¹ These key-value pairs enclosed with curly braces are called [associative arrays](#), and JSON files use syntax for these arrays that is similar to Python dictionaries.

Your first ECM definition

The required information for defining this ECM will be covered in the same order as the *list of parameters* in the *Analysis Approach* section. For all of the fields in the ECM definition, details regarding acceptable values, structure, and formatting are provided in the *ECM JSON schema*.

If after completing this tutorial you feel that you would benefit from looking at additional ECM definitions, you can browse the [ECM definition JSON files](#) available on GitHub.

For this example, we will be creating an ECM for LED troffers for commercial buildings. Troffers are square or rectangular light fixtures designed to be used in a modular dropped ceiling grid commonly seen in offices and other commercial spaces.

The finished ECM specification is available to [download](#) for reference.

To begin, the ECM should be given a descriptive name less than 40 characters long (including spaces). Details regarding the characteristics of the technology that will be included elsewhere in the ECM definition, such as the cost, efficiency, or lifetime, need not be included in the name. The key for the name is simply `name`.

```
{"name": "LED Troffers"}
```

If the ECM describes a technology currently under development, the name should contain the word “Prospective” in parentheses. If the ECM describes or is derived from a published standard or specification, its version number or year should be included in the name.

Note

In this tutorial, JSON entries will be shown with leading and trailing ellipses to indicate that there is additional data in the ECM definition that appears before and/or after the text of interest.

```
{...
  "key_text": "value",
  ...}
```

Applicable baseline market

The applicable baseline market parameters specify the climate zones, building types, structure types, end uses, fuel types, and specific technologies for the ECM.

The climate zone(s) can be given as a single string, if only one climate zone applies, or as a list if a few climate zones apply. The climate zone entry options are outlined in the *Climate zone* section, and formatting details are in the *applicable section* of the JSON schema. If the ECM is suitable for all climate zones, the shorthand string "all" can be used in place of a list of all of the climate zone names. These shorthand terms are discussed further in the *Baseline market shorthand values* section.

LED troffers can be installed in buildings in any climate zone, and for convenience, the available shorthand term will be used in place of a list of all of the climate zone names.

```
{...
  "climate_zone": "all",
  ...}
```

Building type options include specific residential and commercial building types, given in the *Building type* section, as well as several shorthand terms. A single string, list of strings, or shorthand value(s) are all allowable entries, as indicated in the *bldg_type* field reference.

Though LED troffers are most commonly found in office and other small commercial settings, they are found to some limited extent in most types of commercial buildings. Rather than limiting the ECM to only some building types, the technology field will be used to restrict the applicability of the ECM to only the energy used by lighting types that could be replaced by LED troffers.

```
{...
  "bldg_type": "all commercial",
  ...}
```

ECMs can apply to only new construction, only existing buildings, or all buildings both new and existing. This is specified under the *structure_type* key with the values “new,” “existing,” or “all,” respectively.

LED troffers can be installed in both new construction and existing buildings, thus the “all” shorthand is used.

```
{...
  "structure_type": "all",
  ...}
```

The end use(s) correspond to the major building functions or other energy uses provided by the technology described in the ECM. End uses can be specified as a single string or, if multiple end uses apply, as a list. The acceptable formats for the end use(s) are in the *ECM JSON schema* and the acceptable values are listed in the *End use* ECM reference section.² If the ECM describes a technology that affects the thermal load of a building (e.g., insulation), the end use should be given as “heating” and “cooling” in a list.

The only applicable end use for LED troffers is lighting. Changing from fluorescent bulbs typically found in troffers to LEDs will reduce the heat output from the fixture, thus reducing the cooling load and increasing the heating load for the building. These changes in heating and cooling energy use that arise from changes to lighting systems in commercial buildings are accounted for automatically in the energy use calculations for the ECM.

```
{...
  "end_use": "lighting",
  ...}
```

The fuel type generally corresponds to the energy source used for the technology described by an ECM – natural gas for a natural gas heat pump and electricity for an air-source heat pump, for example. The fuel type should be consistent with the end use(s) already specified, based on the *end use reference tables*. Fuel types are listed in the *Fuel type* ECM reference section, and can be specified as a single entry or a list if multiple fuel types are relevant (as indicated in the *ECM JSON schema*). If the ECM describes a technology that affects the thermal load of a building (e.g., insulation), the fuel type should be given as “all” because heating and cooling energy from all fuel types could be affected by those types of technologies.

In the case of LED troffers, electricity is the only relevant fuel type.

```
{...
  "fuel_type": "electricity",
  ...}
```

The technology field drills down into the specific technologies or device types that apply to the end use(s) for the ECM. In some cases, an ECM might be able to replace the full range of incumbent technologies in its end use categories, while in others, only specific technologies might be subject to replacement. As indicated in the *ECM JSON schema*, applicable technologies can be given as a single string, a list of technology names, or using *shorthand values*. If applicable, a technology list can also be specified with a mix of shorthand end use references (e.g., “all lighting”) and specific technology names, such as [“all heating”, “F28T8 HE w/ OS”, “F28T8 HE w/ SR”].

All of the technology names are listed by building sector (residential or commercial) and technology type (supply or demand) in the *relevant section* of the *ECM Definition Reference*. In general, the *residential* and *commercial* thermal

² The end uses also largely correspond to the *residential* and *commercial* end uses specified in the AEO.

load components are the technology names for demand-side energy use, and are relevant for ECMs that apply to the building envelope or windows. Technology names for supply-side energy use generally correspond to major equipment types used in the AEO³ and are relevant for ECMs that are describing those types of equipment within a building.

For this example, LED troffers are likely to replace linear fluorescent bulbs, the typical bulb type in troffers. There are many lighting types for commercial buildings, but we will include all of the lighting types that are specified as T__F__, which correspond to linear fluorescent bulb types, including those with additional modifying text.

```
{...
  "technology": ["T5 F28", "T8 F28 High-efficiency/High-Output", "T8 F32 Commodity", "T8_
↪F59 High Efficiency", "T8 F59 Typical Efficiency", "T8 F96 High Output"],
  ...}
```

Market entry and exit year

The market entry year represents the year the technology is or will be available for purchase and installation. Some ECMs might be prospective, representing technologies not currently available. Others might represent technologies currently commercially available. The market entry year should reflect the current status of the technology described in the ECM. Similarly, the market exit year represents the year the technology is expected to be withdrawn from the market. If the technology described by an ECM will have a lower installed cost or improved energy efficiency after its initial market entry, another ECM should be created that reflects the improved version of the product, and the market exit year should not (in general) be used to force an older technology out of the market.

The market entry year and exit year both require source information. As much as is practicable, a *high quality* reference should be used for both values. If no source is available, such as for a technology that is still quite far from commercialization, a brief explanatory note should be provided for the market entry year source, and the *source_data* fields themselves can be given as null or with empty strings. If it is anticipated that the product will not be withdrawn from the market prior to the end of the model *time horizon*, the exit year and source should be given as null.

LED troffers are currently commercially available with a range of efficiency, cost, and lifetime ratings. It is likely that while LED troffers will not, in general, exit the market within the model *time horizon*, LED troffers with cost and efficiency similar to this ECM are not likely to remain competitive through 2040. It will, however, be left to the analysis to determine whether more advanced lighting products enter the market and supplant this ECM, rather than specifying a market exit year.

```
{...
  "market_entry_year": 2015,
  "market_entry_year_source": {
    "notes": "The source suggests that technologies described in the document are_
↪available on the market by its release date.",
    "source_data": [{
      "title": "High Efficiency Troffer Performance Specification, Version 5.0",
      "author": "",
      "organization": "U.S. Department of Energy",
      "year": 2015,
      "pages": null,
      "URL": "https://betterbuildingssolutioncenter.energy.gov/sites/default/files/
↪attachments/High%20Efficiency%20Troffer%20Performance%20Specification.pdf"]}],
  "market_exit_year": null,
  "market_exit_year_source": null,
  ...}
```

³ Note that this document does not cover lighting, where varying bulb types are used, or Miscellaneous Electric Loads (MELs), which are not broken into specific technologies in the Annual Energy Outlook.

Energy efficiency

The energy efficiency of the ECM must be specified in three parts: the quantitative efficiency (only the value(s)), the units of the efficiency value(s) provided, and source(s) that support the indicated efficiency information. Each of these parameters is specified in a separate field.

The units specified are expected to be consistent with the units for each end use outlined in the *ECM Definition Reference* section.

The source(s) for the efficiency data should be credible sources, such as *those outlined* in the *Analysis Approach* section. The source information should be provided using only the fields shown in the example and should include sufficient information so that the value(s) can be quickly identified from the sources listed. Additional detail regarding the acceptable form for entries in the source are linked to from the *source_data* entry in the ECM JSON schema.

For the example of LED troffers, all lighting data should be provided in the units of lumens per Watt (denoted “lm/W”). LED troffers efficiency information is based on the *High Efficiency Troffer Performance Specification*.

```
{...
  "energy_efficiency": 120,
  "energy_efficiency_units": "lm/W",
  "energy_efficiency_source": {
    "notes": "Initial efficiency value taken from source section II.a.2.a. Efficiency.
    ↳ value increased slightly based on efficacy values for fixtures categorized as '2x4
    ↳ Luminaires for Ambient Lighting of Interior Commercial Spaces' in the DesignLights
    ↳ Consortium Qualified Products List (https://www.designlights.org/qpl).",
    "source_data": [{
      "title": "High Efficiency Troffer Performance Specification, Version 5.0",
      "author": "",
      "organization": "U.S. Department of Energy",
      "year": 2015,
      "pages": 5,
      "URL": "https://betterbuildingssolutioncenter.energy.gov/sites/default/files/
    ↳ attachments/High%20Efficiency%20Troffer%20Performance%20Specification.pdf"
    }]}
  ...}
```

Many additional options exist that enable more complex definitions of energy efficiency, such as incorporating *probability distributions*, providing a *detailed efficiency breakdown* by elements of the applicable baseline market, and using *relative instead of absolute units*. Detailed examples for all of the options are in the *Additional ECM features* section.

Installed cost

The absolute installed cost must be specified for the ECM, including the cost value, units, and reference source. The cost units should be specified according to *the relevant section* of the *ECM Definition Reference*, noting that residential and commercial equipment have different units, and that sensors and controls ECMs also have different units from other equipment types. The source information should be provided using the same keys and structure as the energy efficiency source. For ECMs that describe technologies not yet commercialized, assumptions incorporated into the installed cost estimate should be described in the *notes* section of the source.

If applicable to the ECM, separate cost values can be provided by building type and structure type, as described in the *Detailed input specification* section. Probability distributions can also be used instead of point values for the cost, using the format outlined in the *Probability distributions* section.

For LED troffers, costs are estimated based on an assumption of a single fixture providing 4800 lm, with installation requiring two hours and two people at a fully-burdened cost of \$100/person/hr. The assumptions are articulated using the *notes* key under the *installed_cost_source* key.:

```
{...
  "installed_cost": 233.33,
  "cost_units": "$/1000 lm",
  "installed_cost_source": {
    "notes": "Assumes single fixture provides 4800 lm; requires 2 hour install with 2_
↳ people at a fully-burdened cost of $100/person/hr. Luminaire cost based on a range of_
↳ retail prices found for luminaires with similar specifications found online in October_
↳ 2016.",
    "source_data": [{
      "title": "",
      "author": "",
      "organization": "",
      "year": null,
      "pages": null,
      "URL": ""}]},
  ...}
```

Lifetime

The lifetime of the ECM, or the expected amount of time that the ECM technology will last before requiring replacement, is specified using a structure identical to the installed cost. Again, the lifetime value, units, and source information must be specified for the corresponding keys. The product lifetime can also be specified with a *probability distribution* and/or *different values by building type*. The units should always be in years, ideally as integer values greater than 0. The source information follows the same format as for the energy efficiency and installed cost. For ECMs that describe technologies not yet commercialized, assumptions in the lifetime estimate should be explained in the *notes* section of the source.

LED troffers have rated lifetimes on the order of 50,000 hours, though the [High Efficiency Troffer Performance Specification](#) requires a minimum lifetime of 68,000 hours. The values for lighting lifetimes should be based on assumptions regarding actual use conditions (i.e., number of hours per day), and the *notes* value in the source specification should include that assumption. The LED troffers in this example are assumed to operate 12 hours per day.

```
{...
  "product_lifetime": 15,
  "product_lifetime_units": "years",
  "product_lifetime_source": {
    "notes": "Calculated from 68,000 hrs, stated as item II.c.i, assuming 12 hr/day_
↳ operation.",
    "source_data": [{
      "title": "High Efficiency Troffer Performance Specification, Version 5.0",
      "author": "",
      "organization": "U.S. Department of Energy",
      "year": 2015,
      "pages": 5,
      "URL": "https://betterbuildingssolutioncenter.energy.gov/sites/default/files/_
↳ attachments/High%20Efficiency%20Troffer%20Performance%20Specification.pdf"}]},
  ...}
```

Other fields

The *measure_type* field indicates whether an ECM directly replaces the service of an existing device or building component or improves the efficiency of an existing technology. Examples include a cold-climate heat pump replacing existing electric heating and cooling systems and a window film that decreases solar heat gain, respectively. Further

discussion of how to use the *measure_type* field and illustrative examples are in the *Add-on type ECMs* section.

LED troffers would replace existing troffers that use linear fluorescent bulbs, providing an equivalent building service (lighting) using less energy. The LED troffers ECM is thus denoted as “full service.”

```
{...
  "measure_type": "full service",
  ...}
```

If the ECM is intended to replace baseline technologies of a different technology and/or fuel type, the technologies and fuel types that it switches away from are specified in the *technology* and *fuel_type* fields, respectively, and the technology and fuel type that the ECM switches to are specified in the *tech_switch_to* and *fuel_switch_to* fields, respectively. These fields are explained further, with illustrative examples in the *Technology and/or fuel switching* section. When not applicable, the *tech_switch_to* and *fuel_switch_to* fields should be given the value null.

The LED troffers ECM switches away from linear fluorescent bulbs but does not switch fuels.

```
{...
  "fuel_switch_to": null,
  "tech_switch_to": "LED",
  ...}
```

If the ECM applies to only a portion of the energy use in an applicable baseline market, even after specifying the particular end use, fuel type, and technologies that are relevant, a scaling value can be added to the ECM definition to specify what fraction of the applicable baseline market is truly applicable to that ECM.

When creating a new ECM, it is important to carefully specify the applicable baseline market to avoid the use of the market scaling fraction parameter, if at all possible. If the scaling fraction is not used, the value and the source should be set to null. Details regarding the use of the market scaling fraction can be found in the *Market scaling fractions* section.

No market scaling fraction is required for the LED troffers ECM.

```
{...
  "market_scaling_fractions": null,
  "market_scaling_fractions_source": null,
  ...}
```

Two keys are provided for ECM authors to provide additional details about the technology specified. The *_description* field should include a one to two sentence description of the ECM, including additional references for further details regarding the technology if it is especially novel or unusual. The *_notes* field can be used for explanatory notes regarding the technologies that are expected to be replaced by the ECM and any notable assumptions made in the specification of the ECM not captured in another field.

```
{...
  "_description": "LED troffers for commercial modular dropped ceiling grids that are a replacement for the entire troffer luminaire for linear fluorescent bulbs, not a retrofit kit or linear LED bulbs that slot into existing troffers.",
  "_notes": "Energy efficiency is specified for the luminaire, not the base lamp.",
  ...}
```

Basic contact information regarding the author of a new ECM should be added to the fields under the *_added_by* key.

```
{...
  "_added_by": {
    "name": "Carmen Sandiego",
```

(continues on next page)

(continued from previous page)

```

"organization": "Super Appliances, Inc.",
"email": "carmen.sandiego@superappliances.com",
"timestamp": "2015-07-14 11:49:57 UTC"},
...}

```

When updating an existing ECM, the identifying information for the contributor should be provided in the *_updated_by* field instead of the “_added_by” field. If the ECM is new, the child keys in the “_updated_by” section should be given null values.

```

{...
  "_updated_by": {
    "name": null,
    "organization": null,
    "email": null,
    "timestamp": null},
  ...}

```

The LED troffers ECM that you’ve now written can be simulated with Scout by following the steps in subsequent tutorials. Many technologies will have ECM definitions like the one you just created, but some technologies, like sensors and controls and windows and opaque envelope products, will have definitions that are subtly different. Sensors and controls ECMs augment the efficiency of existing equipment in the stock, rather than replacing existing and supplanting new equipment. To get a feel for what these types of add-on technologies look like as an ECM, you can download and review an Automated Fault Detection and Diagnosis (AFDD) ECM. Additional information about sensors and controls ECMs can be found in the *Add-on type ECMs* section. Windows and opaque envelope technologies reduce demand for heating and cooling instead of increasing the efficiency of the supply of heating and cooling. An ECM for the ENERGY STAR windows version 6 specification is available to download to illustrate demand-reducing ECMs.

Additional ECM features

There are many ways in which an ECM definition can be augmented, beyond the basic example already presented, to more fully characterize a technology. The subsequent sections explain how to implement the myriad options available to add more detail and complexity to your ECMs. Links to download example ECMs that illustrate the feature described are included in each section.

Technology and/or fuel switching

Example – Heat Pump Water Heater (*Details*)

Some ECMs switch a comparable baseline technology to a different technology and/or fuel type. Air or ground source heat pumps, for example, can replace the service of fossil-fired heating systems (different fuel, different technology), or these heat pumps can replace the service of electric resistance baseboard heaters or furnaces (same fuel, different technology). The same goes for heat pump water heaters and for electric ranges and dryers. The *tech_switch_to* field, used in conjunction with the *technology* field and the *fuel_type* and *fuel_switch_to* fields in the baseline market, enables an ECM to replace the service of a different baseline technology and/or fuel type.

To configure such ECMs, the *technology* and *fuel_type* fields should be populated with a list of the technologies and fuel types that, for the applicable end uses, are replaced by the ECM technology. If the ECM is able to replace all technologies for a given fuel type and end use, the *technology* field can be specified using the "all" shorthand value. The *tech_switch_to* field should be set to the most appropriate value for the ECM from Table 1, while the *fuel_switch_to* field should be set to ECM fuel type being switched to.

Table 1: Technology switch to values for various ECM types.

ECM Technology Switched To	JSON Value
Air source heat pump	"ASHP"
Ground source heat pump	"GSHP"
Heat pump water heater	"HPWH"
Electric cooking	"Electric cooking"
Electric drying	"Electric drying"
LED	"LED"

For example, an air source heat pump ECM could replace the service of fossil-fired furnaces paired with central air conditioners.

```
{...
  "fuel_type": ["natural gas", "distillate", "electricity"],
  ...
  "technology": ["furnace (NG)", "furnace (distillate)", "central AC"],
  ...
  "fuel_switch_to": "electricity",
  "tech_switch_to": "ASHP",
  ...}
```

Alternatively, an air source heat pump ECM could replace the service of electric resistance furnaces paired with central air conditioning.

```
{...
  "fuel_type": ["electricity"],
  ...
  "technology": ["resistance heat", "central AC"],
  ...
  "fuel_switch_to": null,
  "tech_switch_to": "ASHP",
  ...}
```

Note

The `ecm_prep.py` module checks for and expects *tech_switch_to* information for all fuel switching and LED measures (“LED”, “solid state”, “Solid State”, or “SSL” in name), as well as for heat pump measures (“HP”, “heat pump”, or “Heat Pump” in name) that apply to electric resistance heating or water heating technologies in the baseline. The `ecm_prep.py` execution will error if this expected information is missing. If a measure meeting those criteria is not intended to represent technology switching, the user can suppress this check by setting the measure’s *tech_switch_to* to value `null`.

A residential heat pump water heater is [available to download](#) to illustrate the setup of the *tech_switch_to*, *fuel_switch_to*, *technology*, and *fuel_type* fields to denote, for this particular example, an electric water heater that can replace water heaters of fossil-fired fuel types.

Time sensitive valuation

In certain cases, ECMs might affect baseline energy loads differently depending on the time of day or season, necessitating time sensitive valuation of ECM impacts. Fig. 5 demonstrates three possible types of time sensitive ECM features.

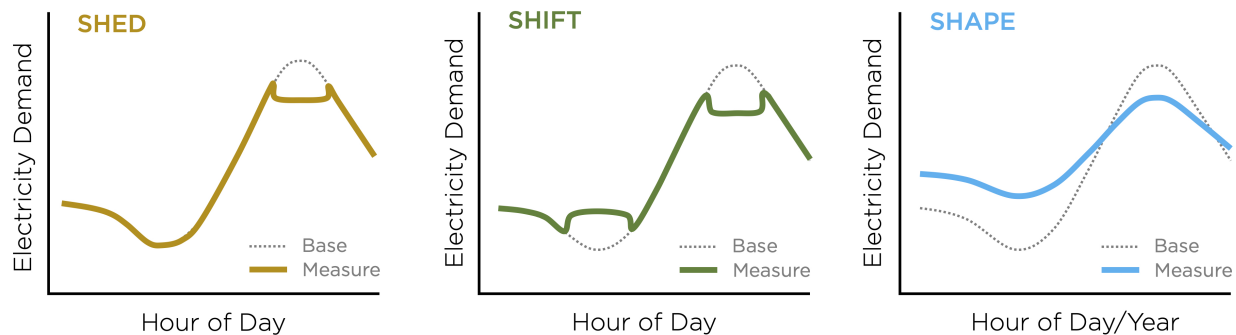


Fig. 5: Time sensitive ECM features include (from left): load shedding, where an ECM reduces load during a certain daily hour range; load shifting, where load is reduced during one daily hour range and increased during another daily hour range; and load shaping, where load may be increased or decreased for any hour of the day/year in accordance with a custom hourly load savings shape.

Such time sensitive ECM features are specified using the *tsv_features* parameter, which adheres to the following general format:

```
{...
  "tsv_features": {
    <time sensitive feature>: {<feature details>},
    ...
  }
```

The *tsv_features* parameter may be broken out by an ECM’s *climate_zone*, *bldg_type*, and/or *end_use*:

```
{...
  "tsv_features": {
    <region 1> : {
      <building type 1> : {
        <end use 1>: {
          <time sensitive feature>: {<feature details>}}}, ...
      ...
    }
  }
```

(continues on next page)

(continued from previous page)

```

    <region N> : {
      <building type N> : {
        <end use N>: {
          <time sensitive feature>: {<feature details>}}}},
    ...}

```

Source information for time sensitive ECM features is specified using the *tsv_source* parameter:

```

{...
  "tsv_source": {
    "notes": <notes>,
    "source_data": [{
      "title": <title>,
      "author": <author>,
      "organization": <organization>,
      "year": <year>,
      "pages": [<start page>, <end page>],
      "URL": <URL>}}],
  ...}

```

The *tsv_source* parameter may be broken out by an ECM's *climate_zone*, *bldg_type*, and/or *end_use*, and by the ECM's time sensitive valuation features:

```

{...
  "tsv_source": {
    <region 1> : {
      <building type 1> : {
        <end use 1>: {
          <time sensitive feature>: {
            "notes": <notes>,
            "source_data": [{
              "title": <title>,
              "author": <author>,
              "organization": <organization>,
              "year": <year>,
              "pages": [<start page>, <end page>],
              "URL": <URL>}}]}}, ...
    <region N> : {
      <building type N> : {
        <end use N>: {
          <time sensitive feature>: {
            "notes": <notes>,
            "source_data": [{
              "title": <title>,
              "author": <author>,
              "organization": <organization>,
              "year": <year>,
              "pages": [<start page>, <end page>],
              "URL": <URL>}}]}},
    ...}

```

Each time sensitive ECM feature is further described below with illustrative example ECMs.

Note

Time sensitive ECM features are currently only supported for ECMs that affect the electric fuel type across the 2019 EIA Electricity Market Module (EMM) regions, and may not be defined as fuel switching measures.

Accordingly, when preparing an ECM with time sensitive features, the user should ensure that:

- 1) the ECM's *fuel_type* parameter is set to "electricity", and the ECM's *fuel_switch_to* parameter is set to null;
- 2) `ecm_prep.py` is executed with the `--alt_regions` option specified; and
- 3) EMM is subsequently selected as the alternate regional breakout.

Users are also encouraged to use the `--site_energy` option when executing `ecm_prep.py` for ECMs with time sensitive features, as utility planners are often most interested in the change in the electricity demand (rather than generation) that may result from ECM deployment.

Note

The effects of an ECM's time sensitive features are applied *on top of* the ECM's static energy efficiency impact on baseline loads, as defined in the ECM's *energy_efficiency* parameter.

Example – Commercial AC (Shed) ECM (Details)

The first type of time sensitive ECM feature sheds (reduces) a certain percentage of baseline electricity demand (defined by the parameter *relative energy change fraction*) during certain days of a *reference year* (defined by the parameters *start_day* and *stop_day*) and hours of the day (defined by the parameters *start_hour* and *stop_hour*.)

```
{...
  "tsv_features": {
    "shed": {
      "relative energy change fraction": 0.1,
      "start_day": 152, "stop_day": 174,
      "start_hour": 12, "stop_hour": 20}},
  ...}
```

In this example, the ECM sheds 10% of electricity demand between the hours of 12–8 PM on all summer days (Jun–Sep, days 152–173 in the *reference year*).

Tip

Two day ranges may be provided by specifying the parameters *start_day* and *stop_day* as lists with two elements:

```
{...
  "start_day": [1, 335],
  "stop_day": [91, 365],
  ...}
```

In this example, the ECM features will be applied to all winter months (Dec, days 335–365; and Jan–Mar, days 1–90 in the *reference year*).

Moreover, if an ECM feature applies to *all* days of the year, the parameters *start_day* and *stop_day* need not be provided.

A commercial load shedding ECM is [available for download](#).

Example – Commercial AC (Load Shift) ECM ([Details](#))

The second type of time sensitive ECM feature shifts baseline energy loads from one time of day to another by redistributing loads reduced during a certain hour range to earlier times of day.

As with the shed feature, the [start_day](#) and [stop_day](#) and [start_hour](#) and [stop_hour](#) parameters are used to determine the day and hour ranges from which to shift the load reductions, respectively. The magnitude of the load reduction is again defined by the [relative energy change fraction](#) parameter. The [offset_hrs_earlier](#) parameter is used to determine which hour range to redistribute the load reductions to.

```
{...
  "tsv_features": {
    "shift": {
      "offset_hrs_earlier": 12,
      "relative energy change fraction": 0.1,
      "start_day": 152, "stop_day": 174,
      "start_hour": 12, "stop_hour": 20},
    ...}
}
```

In this example, the ECM shifts 10% of electricity demand between the hours of 12–8 PM to 12 hours earlier (e.g., to 12–8 AM) on all summer days (Jun–Sep, days 152–173 in the [reference year](#)).

A commercial load shifting ECM is [available for download](#).

Example – Commercial AC (Shape - Custom Daily) ECM ([Details](#))

Example – Commercial AC (Shape - Custom 8760) ECM ([Details](#))

Example – Sample 8760 CSV ([Details](#))

The final type of time sensitive ECM feature applies hourly savings fractions to baseline loads in accordance with a custom savings shape that represents either a typical day or all 8760 hours of the year.

In the first case, custom hourly savings for a typical day are defined in the [custom_daily_savings](#) parameter; the hourly savings are specified as a list with 24 elements, with each element representing the fraction of hourly baseline load that an ECM saves. These hourly savings are applied for each day of the year in the range defined by the [start_day](#) and [stop_day](#) parameters, as for the shed and shift features.

```
{...
  "tsv_features": {
    "shape": {
      "start_day": 152, "stop_day": 174,
      "custom_daily_savings": [
        0.5, 0.5, 0.5, 0.5, 0.5, 0.6, 1, 1.3, 1.4, 1.5, 1.6, 1.8,
        1.9, 2, 1, 0.5, 0.75, 0.75, 0.75, 0.75, 0.5, 0.5, 0.5, 0.5]},
    ...}
}
```

In this example, the ECM reduces hourly loads between 50–200% on all summer days (days 152–174 in the [reference year](#)). Note that savings fractions may be specified as greater than 1 to represent the effects of on-site energy generation on a building's overall load profile.

A commercial daily load shaping ECM is [available for download](#).

In the second case, the custom savings shape represents hourly load impacts for all 8760 hours in the [reference year](#). Here, the measure definition links to a supporting CSV file via the [custom_annual_savings](#) parameter. The CSV is expected to be present in the `./ecm_definitions/energyplus_data/savings_shapes` folder, with one CSV per measure JSON in `./ecm_definitions` that uses this feature.

```
{...
  "tsv_features": {
    "shape": {
      "custom_annual_savings": "sample_8760.csv"
    }
  },
  ...}
```

In this example, the supporting CSV file path is `./ecm_definitions/energyplus_data/savings_shapes/sample_8760.csv`. The CSV file must include the following data (by column name):

- *Hour of Year*. Hour of the simulated year, spanning 1 to 8760. The simulated year must match the [reference year](#) in terms of starting day of the week (Sunday) and total number of days (365).
- *Climate Zone*. Applicable [ASHRAE 90.1-2016 climate zone](#) (see Table 2); currently, only the 14 contiguous U.S. climate zones (2A through 7) are supported.
- *Net Load Version*. This column indicates the one or two representative [EIA Electricity Market Module \(EMM\)](#) net utility system load [profiles](#) for the given climate zone that determine energy flexibility measure characteristics (e.g., targeted shed/shift periods) for that climate zone; this distinction is only relevant to flexibility measures. [Table 2](#) summarizes default periods of net peak and low system demand under the *AEO Low Oil and Gas Supply* side case for each ASHRAE climate zone in the summer (S) and winter (W); the “Version” column of [Table 2](#) indicates cases where two system load profiles are used to define these peak/low demand periods for a given climate zone.

Table 2: Net peak and low system demand periods by ASHRAE climate zone in winter (W) and summer (S), using data from the *AEO 2026 Low Oil and Gas Supply* side case for the year 2050.

Climate	Version	EMM Reg.	Peak (W)	Peak (S)	Low (W)	Low (S)
2A	1	FRCC	5-9PM	5-9PM	10AM-2PM	9AM-1PM
2A	2	MISS	6-10PM	5-9PM	12-4PM	7-11AM
2B	1	SRSG	5-9PM	5-9PM	10AM-2PM	3-7AM
3A	1	SRSE	5-9AM	5-9PM	11AM-3PM	8AM-12PM
3A	2	TRE	5-9PM	5-9PM	1-5PM	8AM-12PM
3B	1	CASO	5-9PM	5-9PM	9AM-1PM	8AM-12PM
3B	2	BASN	5-9PM	5-9PM	11AM-3PM	8AM-12PM
3C	1	CANO	5-9PM	5-9PM	9AM-1PM	9AM-1PM
4A	1	PJME	5-9PM	5-9PM	1-5PM	3-7AM
4A	2	SRCE	5-9AM	5-9PM	11AM-3PM	8AM-12PM
4B	1	SRSG	5-9PM	5-9PM	10AM-2PM	3AM-7AM
4B	2	CANO	5-9PM	5-9PM	9AM-1PM	9AM-1PM
4C	1	NWPP	5-9PM	5-9PM	2-6PM	3-7AM
4C	2	CANO	5-9PM	5-9PM	9AM-1PM	9AM-1PM
5A	1	PJMW	6-10PM	5-9PM	1-5PM	3-7AM
5A	2	MISE	5-9PM	5-9PM	1-5PM	10AM-2PM
5B	1	RMRG	5-9PM	5-9PM	2-6AM	3-7AM
5B	2	CANO	5-9PM	5-9PM	9AM-1PM	9AM-1PM
5C	1	NWPP	5-9PM	5-9PM	2-6PM	3-7AM
6A	1	MISW	1-5AM	5-9PM	12-4PM	2-6AM
6A	2	ISNE	11AM-3PM	12-4PM	12-4AM	2-6AM
6B	1	NWPP	5-9PM	5-9PM	2PM-6PM	3-7AM
6B	2	CASO	5-9PM	5-9PM	9AM-1PM	8AM-12PM
7	1	MISW	1-5AM	5-9PM	12-4PM	2-6AM
7	2	RMRG	5-9PM	5-9PM	2-6AM	3-7AM

- *Building Type*. Applicable EnergyPlus building type; currently supported representative building types are:

- SF ([ResStock](#) single family prototype)
- MF ([ResStock](#) multi family prototype)
- MH ([ResStock](#) mobile home prototype)
- MediumOfficeDetailed or MediumOffice ([DOE Commercial Prototypes](#))
- LargeOfficeDetailed or LargeOffice ([DOE Commercial Prototypes](#))
- LargeHotel ([DOE Commercial Prototypes](#))
- RetailStandalone ([DOE Commercial Prototypes](#))
- Warehouse ([DOE Commercial Prototypes](#))
- *End Use*. Electric end use; currently supported options are:
 - heating
 - cooling
 - lighting
 - water heating
 - refrigeration
 - ventilation
 - drying
 - cooking
 - plug loads
 - dishwasher
 - clothes washing
 - drying
 - pool heaters and pumps
 - fans and pumps
 - other
- *Baseline Load*. Load (for residential, average hourly kW per home across all homes sampled in the representative city for the climate zone; for commercial, hourly kW per prototypical building) for given hour of year, climate zone, net load version, building type, and end use under the baseline case.
- *Measure Load*. Load (for residential, average hourly kW per home across all homes sampled in the representative city for the climate zone; for commercial, hourly kW per prototypical building) for given hour of year, climate zone, net load version, building type, and end use after measure application.
- *Relative Savings*. Calculated as: $(\text{Hourly Measure Load} - \text{Hourly Baseline Load}) / (\text{Total Annual Baseline Load})$.

A commercial 8760 load shaping ECM is [available for download](#); this example ECM is set up to draw from an example 8760 CSV, which is also [available for download](#). Note that to effectively run the [commercial 8760 load shaping ECM](#), the [example 8760 CSV](#) must be moved to the `.ecm_definitions/energyplus_data/savings_shapes` folder.

Example – Commercial AC (Multiple TSV) ECM ([Details](#))

Finally, it is possible to define ECMs that combine multiple time sensitive features at once—e.g., an ECM that turns down the thermostat temperature during early evening hours on winter days (shed) and pre-cools through the mid-day hours while setting up the thermostat temperature during early evening hours on summer days (shift). Such measures are handled by nesting multiple feature types under the `tsv_features` parameter in the ECM definition.

```
{...
  "tsv_features": {
    "shed": {
      "relative energy change fraction": 0.1,
      "start_day": [1, 335], "stop_day": [91, 365],
      "start_hour": 16, "stop_hour": 20},
    "shift": {
      "offset_hrs_earlier": 4,
      "relative energy change fraction": 0.1,
      "start_day": 152, "stop_day": 174,
      "start_hour": 16, "stop_hour": 20}
  }
}
```

In this example, the first feature will represent baseline load shedding between the hours of 4–8 PM on all winter days, while the second feature will shift baseline loads occurring between 4–8 PM to the hours of 12–4 PM on all summer days.

A commercial load shedding and shifting ECM is *available for download*.

Baseline market shorthand values

Example – Whole Building Sub-metering ECM (*Details*)

If an ECM applies to multiple building types, end uses, or other applicable baseline market categories⁴, the specification of the baseline market and, in some cases, other fields, can be greatly simplified by using shorthand strings. When specifying the applicable baseline market, for example, an ECM might represent a technology that can be installed in any residential building, indicated with the “all residential” string for the building type key.

```
{...
  "bldg_type": "all residential",
  ...}
```

Similarly, an ECM that applies to any climate zone can use “all” as the value for the climate zone key.

```
{...
  "climate_zone": "all",
  ...}
```

These shorthand terms, when they encompass only a subset of the valid entries for a given field (e.g., “all commercial,” which does not include any residential building types), can also be mixed in a list with other valid entries for that field.

```
{...
  "bldg_type": ["all residential", "small office", "lodging"],
  ...}
```

The *ECM definition reference* specifies whether these shorthand terms are available for each of the applicable baseline market fields and what shorthand strings are valid for each field.

If these shorthand terms are used to specify the applicable baseline market fields, the energy efficiency, installed cost, and lifetime may be specified with a single value. For example, if an ECM applies to “all residential,” “small office,” and “lodging” building types, they could all share the same installed cost.

⁴ The “applicable baseline market” is comprised of the *climate_zone*, *bldg_type*, *structure_type*, *fuel_type*, *end_use*, and *technology* fields.

```
{...
  "bldg_type": ["all residential", "small office", "lodging"],
  ...
  "installed cost": 5825,
  ...}
```

Alternately, a *detailed input specification* for energy efficiency, installed cost, or lifetime can be used. Using the same building types example, if a detailed input specification is used for the installed cost, a cost value must be given for *all* of the specified building types.

```
{...
  "installed_cost": {
    "all residential": 5530,
    "small office": 6190,
    "lodging": 6015},
  ...}
```

Again using the same example, separate installed costs can also be specified for each of the residential building types, even if they are indicated as a group in the building type field using the “all residential” shorthand.

```
{...
  "installed_cost": {
    "single family home": 5775,
    "multi family home": 5693,
    "mobile home": 5288,
    "small office": 6190,
    "lodging": 6015},
  ...}
```

A whole building sub-metering ECM is [available for download](#) that illustrates the use of shorthand terms by employing the “all” shorthand term for most of the applicable baseline market fields (*climate_zone*, *bldg_type*, *structure_type*, *fuel_type*, *end_use*, and *technology*) and the “all commercial” shorthand term as one of the building types and to define separate installed costs for the various building types that apply to the ECM. If you would like to see additional examples, many of the other examples available to download in this section use shorthand terms for one or more of their applicable baseline market fields.

Detailed input specification

Example – Thermoelastic Heat Pump ECM (*Details*)

The energy efficiency, installed cost, and lifetime values in an ECM definition can be specified as a point value or with separate values for one or more of the following applicable baseline market keys: *climate_zone*, *bldg_type*, *end_use* and *structure_type*. As shown in [Table 3](#), the allowable baseline market keys are different for the energy efficiency, installed cost, and lifetime values.

Table 3: The baseline market keys that can be used to provide a detailed specification of the energy efficiency, installed cost, or lifetime input fields in an ECM definition depend on which field is being specified.

Baseline Market Key	Energy Efficiency	Installed Cost	Product Lifetime
<i>climate_zone</i>	X	X	
<i>bldg_type</i>	X	X	X
<i>end_use</i>	X		
<i>structure_type</i>	X	X	

A detailed input specification for any of the fields should consist of a dict with keys from the desired baseline market field(s) and the appropriate values given for each key. For example, an HVAC-related ECM, such as a central AC unit, will generally have efficiency that varies by *climate zone*, which can be captured in the energy efficiency input specification.

```
{...
  "energy_efficiency": {
    "AIA_CZ1": 1.6,
    "AIA_CZ2": 1.54,
    "AIA_CZ3": 1.47,
    "AIA_CZ4": 1.4,
    "AIA_CZ5": 1.28},
  ...}
```

Tip

Detailed input specifications for ECM energy efficiency and installed cost may follow a different regional breakout than what is reflected in the ECM's *climate zone* attribute so long as the breakouts conform to one of either the *AIA* or *IECC climate regions*. If IECC climate zones are used for the breakouts, breakout keys should use the following format: IECC_CZ1, IECC_CZ2, ... IECC_CZ8.

Note

If a detailed input specification is used, all of the applicable baseline market keys *must* be given and have a corresponding value. For example, an ECM that applies to three building types and has a detailed input specification for installed cost must have a cost value for all three building types. (Exceptions may apply if alternate regional breakouts of performance or cost are used, as described in the previous tip, or if the partial shortcuts “all residential” and “all commercial” are used – see the *baseline market shorthand values* documentation.)

ECMs that describe technologies that perform functions across multiple end uses will necessarily require an energy efficiency definition that is specified by fuel type. Air-source heat pumps, which provide both heating and cooling, are an example of such a technology.

```
{...
  "energy_efficiency": {
    "heating": 1.2,
    "cooling": 1.4},
  ...}
```

For an ECM that applies to both new and existing buildings, the installed cost might vary for those *structure types* due to differences in the number of labor hours required to install the technology as a building is being constructed versus an installation that begins with teardown and requires more careful cleanup and management of dust and noise.

```
{...
  "installed_cost": {
    "new": 26,
    "existing": 29},
  ...}
```

If a detailed input specification includes two or more baseline market keys, the keys should be placed in a nested dict structure adhering to the following key hierarchy: *climate_zone*, *bldg_type*, *end_use* and *structure_type*. Multi-function heat pumps, which provide heating, cooling, and water heating services, are an example of a case where a

detailed energy efficiency specification by climate zone and end use might be appropriate.

```
{...
  "energy_efficiency": {
    "AIA_CZ1": {
      "heating": 1.05,
      "cooling": 1.3,
      "water heating": 1.25},
    "AIA_CZ2": {
      "heating": 1.15,
      "cooling": 1.26,
      "water heating": 1.31},
    "AIA_CZ3": {
      "heating": 1.3,
      "cooling": 1.21,
      "water heating": 1.4},
    "AIA_CZ4": {
      "heating": 1.4,
      "cooling": 1.16,
      "water heating": 1.57},
    "AIA_CZ5": {
      "heating": 1.4,
      "cooling": 1.07,
      "water heating": 1.7}},
  ...}
```

If an input has a detailed specification, the units need not be given in an identical dict structure. The units can be specified using the simplest required structure, including as a single string, while matching the required units specified for *energy efficiency* and *installed cost*. Product lifetime units can always be given as a single string since all lifetime values should be in years. For the first example, energy efficiency units will not vary across climate zones.

```
{...
  "energy_efficiency_units": "COP",
  ...}
```

Similarly, in the third example, installed cost units for a given technology will not vary by building structure type.

```
{...
  "installed_cost_units": "$/ft^2 wall",
  ...}
```

In the second example, while the energy efficiency units are generally different for each end use, the *energy efficiency units reference* shows that heating (from a heat pump) and cooling have the same units, thus the units do not need to be specified by end use for this particular case.

```
{...
  "energy_efficiency_units": "COP",
  ...}
```

In the fourth example, where the energy efficiency is specified by climate zone and end use, the units will only vary by end use, thus the units dict does not need to be identical in structure to the energy efficiency dict, and can be specified using only the end uses.

```
{...
  "energy_efficiency_units": {
    "heating": "COP",
    "cooling": "COP",
    "water heating": "UEF"},
  ...}
```

While all of the examples shown use absolute units, *relative savings values* and *probability distributions* can also be used with detailed input specifications. If an ECM can be described using one or more *shorthand terms*, these strings can be used as keys for a detailed input specification; this ability is particularly helpful when using the “all residential” and/or “all commercial” building type shorthand strings.

When a detailed input specification is given, the corresponding source information need not be specified in the same type of nested dict format, particularly if all of the data are drawn from a single source. Even if multiple sources are required, all of the sources may be given with separate dicts in a single list under the *source_data* key, along with an explanation of what data are drawn from each source given in the *notes* field.

Finally, any ECM that includes one or more detailed input specifications should have some discussion of the detailed specification and any underlying assumptions included in either the *_notes* field in the JSON or the *notes* field for the source information for each detailed input specification. As the complexity of the specification increases, the detail of the explanation should similarly increase.

A thermoelastic heat pump ECM is [available for download](#) to illustrate the use of the detailed input specification approach for the installed cost data and units, as well as the page information for the installed cost source.

Relative energy efficiency units

Example – Occupant-centered Controls ECM (*Details*)

In addition to the absolute units used in the *initial example*, any ECM can have energy efficiency specified with the units “relative savings (constant)” or “relative savings (dynamic)”. In either case, the energy efficiency value should be given as a decimal value between 0 and 1, corresponding to the percentage improvement from the baseline (i.e., the existing stock) – a value of 0.2 corresponds to a 20% energy savings relative to the baseline.

Note

Absolute efficiency units are preferred (except for sensors and controls ECMs). Absolute units are more commonly reported in test results and product specifications. In addition, using relative savings leaves some uncertainty regarding whether there are discrepancies between the baseline used to calculate the savings percentage and the baseline in Scout.

When the units are “relative savings (constant),” the value that is given is assumed to be the same in every year, independent of improvement in the efficiency of technologies comprising the baseline. That is, an “energy_efficiency” value of 0.3 with the units “relative savings (constant)” means that the ECM will achieve a 30% reduction in energy use compared to the baseline in the current year and a 30% reduction in energy use compared to the baseline in all future years.

If “relative savings (dynamic)” is used, the percentage savings are reduced in future years to account for efficiency improvements in the baseline. These reductions are calculated relative to an anchor year, which is the year for which the specified savings percentage was calculated. The anchor year is specified as an integer along with the units string in a list. (In the example shown, 2014 is the anchor year.)

```
{...
  "energy_efficiency_units": ["relative savings (dynamic)", 2014],
  ...}
```

Relative units can be combined with *detailed input specifications*.

```
{...
  "energy_efficiency": {
    "AIA_CZ1": 0.13,
    "AIA_CZ2": 0.127,
    "AIA_CZ3": 0.123,
    "AIA_CZ4": 0.118,
    "AIA_CZ5": 0.11},
  "energy_efficiency_units": "relative savings (constant)",
  ...}
```

If appropriate for a given ECM, absolute and relative units can also be mixed in a *detailed input specification*.

```
{...
  "energy_efficiency": {
    "heating": 1.2,
    "cooling": 0.25},
  "energy_efficiency_units": {
    "heating": "COP",
    "cooling": ["relative savings (dynamic)", 2016]},
  ...}
```

An occupant-centered controls ECM *available for download*, like all controls ECMs, uses relative savings units. It also illustrates several other features discussed in this section, including *detailed input specification*, and the *add-on measure type*.

Market scaling fractions

Example – Automated Fault Detection and Diagnosis ECM (*Details*)

If an ECM applies to only a portion of the energy use in an applicable baseline market, even after specifying the particular building type, end use, fuel type, and technologies that are relevant, the market scaling fraction can be used to specify the fraction of the applicable baseline market that is truly applicable to that ECM. The market scaling fraction thus reduces the size of all or a portion of the applicable baseline market beyond what is achievable using only the baseline market fields. All scaling fraction values should be between greater than 0 and less than 1, where a value of 0.4, for example, indicates that 40% of the baseline market selected applies to that ECM.

Note

When creating a new ECM, it is important to carefully specify the applicable baseline market to avoid the use of the market scaling fraction parameter, if at all possible.

Since the scaling fraction is not derived from the EIA data used to provide a common baseline across all ECMs in Scout, source information must be provided, and it is especially important that the source information be correct and complete. The market scaling fraction source information should be supplied as a dict corresponding to a single source. If multiple values derived from multiple sources are reported, source information can be provided using the same nested dict structure as the scaling fractions themselves. The source field for the market scaling fraction has keys similar to those under the “source_data” key associated with other ECM data, but with an additional *fraction_derivation* key. The fraction derivation is a string that should include an explanation of how the scaling value(s) are calculated from the source(s) given.

When *preparing the ECM for analysis*, if a scaling fraction is specified, the source fields are automatically reviewed to ensure that either a) a “title,” “author,” “organization,” and “year” are specified or b) a URL from an acceptable

source⁵ is provided. While these are the minimum requirements, the source information fields should be filled out as completely as possible. Additionally, the “fraction_derivation” field is checked for the presence of some explanatory text. If any of these required fields are missing, the ECM will not be included in the *prepared ECMs*.

As an example, for a multi-function fuel-fired heat pump ECM for commercial building applications, if the system is to provide space heating and cooling and water heating services, it is most readily installed in a building that already has some non-electric energy supply. If it is assumed that any building with a non-electric heating system would be a viable installation target for this technology, market scaling fractions can be applied to restrict the baseline market to correspond with that assumption.

```
{...
  "market_scaling_fractions": {
    "cooling": 0.53,
    "water heating": 0.53},
  "market_scaling_fractions_source": {
    "title": "2012 Commercial Buildings Energy Consumption Survey (CBECS) Public Use_
↪Microdata",
    "author": "U.S. Energy Information Administration (EIA)",
    "organization": "",
    "year": "2016",
    "URL": "http://www.eia.gov/consumption/commercial/data/2012/index.php?view=microdata",
    "fraction_derivation": "Assuming that only buildings with natural gas or propane_
↪heating can be retrofitted with a multi-function fuel-fired heat pump, 53.1% of_
↪commercial building floor space in CBECS is from buildings with natural gas or propane_
↪primary heating systems."},
  ...}
```

As shown in the example, if the ECM applies to multiple building types, climate zones, or technologies, for example, different scaling fraction values can be supplied for some or all of the baseline market. The method for specifying multiple scaling fraction values is similar to that outlined in the *Detailed input specification* sub-section. This detailed breakdown of the market scaling fraction can only include keys that are included in the applicable baseline market. For example, if the applicable baseline market includes only residential buildings, no commercial building types should appear in the market scaling fraction breakdown. If all residential buildings are in the applicable baseline market, however, the market scaling fractions can be separately specified for each residential building type.

The automated fault detection and diagnosis (AFDD) ECM *available for download* illustrates the use of the market scaling fraction to limit the applicability of the ECM to only buildings with building automation systems (BAS), since that is a prerequisite for the implementation of the AFDD technology described in the ECM.

Add-on type ECMs

Example – Plug-and-Play Sensors ECM (*Details*)

Technologies that affect the operation of or augment the efficiency of the existing components of a building must be defined differently in an ECM than technologies that replace a building component. Examples include sensors and control systems, window films, and daylighting systems. These technologies improve or affect the operation of another building system – HVAC or other building equipment, windows, and lighting, respectively – but do not replace those building systems.

For these technologies, several of the fields of the ECM must be configured slightly differently. First, the applicable baseline market should be set for the end uses and technologies that are affected by the technology, not those that describe the technology. For example, an automated fault detection and diagnosis (AFDD) system that affects heating and cooling systems should have the end uses “heating” and “cooling,” not some type of electronics or miscellaneous electric load (MEL) end use. Second, the energy efficiency values should have *relative savings* units and the installed

⁵ Acceptable domains include eia.gov, doe.gov, energy.gov, data.gov, energystar.gov, epa.gov, census.gov, pnnl.gov, lbl.gov, nrel.gov, sciencedirect.com, costar.com, and navigantresearch.com.

cost units should match those specified in the [ECM Definition Reference](#), noting that they are different for sensors and controls ECMs. Finally, the `measure_type` field should have the value "add-on" instead of "full service".

A plug-and-play sensors ECM is [available to download](#) to illustrate the use of the “add-on” ECM type.

ECM-specific retrofit rate

Example – LED Troffers (High Retrofit Rate) ([Details](#))

Certain ECMs may be targeted towards accelerating typical equipment retrofit rates - e.g., a persistent information campaign that improves consumer awareness of available incentives for replacing older appliances with ENERGY STAR alternatives. Alternatively, a user may simply wish to explore the sensitivity of ECM outcomes to variations in Scout’s default equipment retrofit rate.⁶

To configure such ECMs, the optional `retro_rate` field should be populated with a point value between 0 and 1 that represents the assumed retrofit rate for the ECM. For example, if an ECM is assumed to increase the rate of existing technology stock retrofits to 10% of the existing stock, this effect would be represented as follows.

```
{...
  "retro_rate": 0.1,
  ...}
```

Alternatively, the user may place a probability distribution on this rate - see [Probability distributions](#) for more details.

Supporting source information for the ECM-specific retrofit rate should be included in the ECM definition using the `retro_rate_source` field.

A second version of the [LED troffer example](#) that assumes a higher retrofit rate (10%) is [available to download](#).

Probability distributions

Example – LED Bulbs ECM ([Details](#))

Probability distributions can be added to the installed cost, energy efficiency, and lifetime specified for ECMs to represent uncertainty or known, quantified variability in one or more of those values. In a single ECM, a probability distribution can be applied to any one or more of these parameters. Probability distributions cannot be specified for any other parameters in an ECM, such as the market entry or exit years, market scaling fractions, or to either the energy savings increase or cost reduction parameters in [package ECMs](#).

Where permitted, probability distributions are specified using a list. The first entry in the list identifies the desired distribution. Subsequent entries in the list correspond to the required and optional parameters that define that distribution type, according to the [numpy.random module documentation](#), excluding the optional “size” parameter.⁷ The `uniform`, `normal`, `lognormal`, `triangular`, `weibull`, and `gamma` distributions are currently supported. (Note that the normal and log-normal distributions’ scale parameter is standard deviation, not variance.)

For a given ECM, if the installed cost is known to vary uniformly between 1585 and 2230 \$/unit, that range can be specified with a probability distribution.

```
{...
  "installed_cost": ["uniform", 1585, 2230],
  ...}
```

Probability distributions can be specified in any location in the energy efficiency, installed cost, or product lifetime specification where a point value would otherwise be used. Distributions do not have to be provided for every value in

⁶ The retrofit rate assumption only affects the [Maximum Adoption Potential](#) scenario results, in which realistic equipment turnover dynamics are considered.

⁷ The size parameter specifies the number of samples to draw from the specified distribution. The number of samples is preset to be the same for all ECMs to ensure consistency.

a detailed specification if it is not relevant or there are insufficient supporting data. Different distributions can be used for each value if so desired.

```
{...
  "energy_efficiency": {
    "heating": ["normal", 2.3, 0.4],
    "cooling": ["lognormal", 0.9, 0.2],
    "water heating": 1.15},
  ...}
```

An ENERGY STAR LED bulbs ECM is [available for download](#) to illustrate the use of probability distributions, in that case, on installed cost and product lifetime.

Technology diffusion

Example – Commercial Heat Pump Water Heater ECM ([Details](#))

Technology diffusion models describe how a given technology spreads into the market. Between its market entry and exit year, a technology can have a changing adoption rate to reflect changes in market conditions or consumer awareness. This adoption rate can be modeled in one of two ways.

For a given ECM, the diffusion model can be expressed as a series of fractions (between 0 and 1) for one or more years:

```
{...
  "diffusion": {
    "fraction_2020": '0.3',
    "fraction_2030": '0.5',
    "fraction_2040": '1'},
  ...}
```

These diffusion fractions can be defined for any year between the market entry and exit year. For years without a specified fraction, a diffusion fraction will be derived through linear interpolation. The number of diffusion fractions specified can range from one to the number of years between the market entry and exit year.

Alternatively, the diffusion curve can be expressed through the parameters p and q of the Bass model:

```
{...
  "diffusion": {
    "bass_model_p": '0.001645368',
    "bass_model_q": '1.455182'},
  ...}
```

If no diffusion parameter is provided, or if it is provided in a format different than the two formats listed above, the diffusion value will default to 1 for all years between the market entry and exit year.

A commercial heat pump water heater ECM is [available for download](#) to illustrate the use of the technology diffusion parameters.

Editing existing ECMs

All of the ECM definitions are stored in the `./ecm_definitions` folder. To edit any of the existing ECMs, open that folder and then open the JSON file for the ECM of interest. Make any desired changes, save, and close the edited file. Like new ECMs, all edited ECMs must be prepared following the steps in [Tutorial 3](#).

Making changes to the existing ECMs will necessarily overwrite previous versions of those ECMs. If both the original and revised version of an ECM are desired for subsequent analysis, make a copy of the original JSON file (copy and paste the file in the same directory) and rename the copied JSON file with an informative differentiating name. When

revising the copied JSON file with the new desired parameters, take care to ensure that the ECM name is updated as well.

Tip

No two ECMs can share the same file name *or* name given in the JSON.

Creating and editing package ECMs

Package ECMs are not actually unique ECMs, rather, they are combinations of existing (single technology) ECMs specified by the user. Existing ECMs can be included in multiple different packages; there is no limit to the number of packages to which a single ECM may be added. There is also no limit on the number of ECMs included in a package.

Currently, the ECM packaging capability is oriented around combinations of HVAC equipment, windows and envelope, and/or controls ECMs, as well as around combinations of lighting equipment and controls ECMs. Users attempting to package unsupported types of ECMs will receive an error message that informs them of the types of ECMs that the packaging capability is meant to support.

Note

When HVAC equipment and windows and envelope (W/E) ECMs are included together in a package, the W/E costs will be excluded from the overall package costs by default. This is necessary to match the nature of the packaged HVAC + W/E measure's installed costs with that of Scout's underlying technology competition model, which is developed around HVAC equipment costs. Nevertheless, W/E costs can be included for such packages by specifying the `--pkg_env_costs` command line option described in [Additional preparation options](#).

Package ECMs are specified in the `package_ecms.json` file, located in the `./ecm_definitions` folder. A version of the `package_ecms.json` file with a single blank ECM package definition is available for [download](#).

In the package ECMs JSON definition file, each ECM package is specified in a separate dict with three keys: `name`, `contributing_ECMs`, and `benefits`. The package name should be a unique name (from other packages and other individual ECMs). The `contributing_ECMs` should be a list of the ECM names to include in the package, separated by commas. The individual ECM names should match exactly with the `name` field in each of the ECM's JSON definition files.

Packaging ECMs may result in integrative improvements in energy use and/or reductions in total installed cost that may be considered via the packaged ECM's `benefits` attribute. Information under this attribute is specified in a dict with three keys, `energy_savings_increase`, `cost_reduction` and `source`. The `energy_savings_increase` and `cost_reduction` values should be fractions between 0 and 1 (in general) representing the percentage savings or cost changes. The `energy_savings_increase` can be assigned a value greater than 1, indicating an increase in energy savings of greater than 100%, but robust justification of such a significant improvement should be provided in the source information. If no benefits are relevant for one or both keys, the values can be given as `null` or `0`. The source information for the efficiency or cost improvements are provided in a nested dict structure under the `source` key. The source information should have the same structure as in individual ECM definitions. This structure for a single package ECM that incorporates three ECMs and yields a cost reduction of 15% over the total for those three ECMs is then:

```
{
  "name": "First package name",
  "contributing_ECMs": ["ECM 1 name", "ECM 2 name", "ECM 3 name"],
  "benefits": {
    "energy_savings_increase": 0,
    "cost_reduction": 0.15,
    "source": {
      "notes": "Information about how the indicated benefits value(s) were derived.",
      "source_data": [
        {
          "title": "The Title",
          "author": "Source Author",

```

(continues on next page)

(continued from previous page)

```

    "organization": "Organization Name",
    "year": "2016",
    "pages": "15-17"}]
  }
}
```

All of the intended packages should be specified in the package_ecms.json file. For example, the contents of the file should take the following form if there are three desired packages, with three, two, and four ECMs, respectively.

```

[{"name": "First package name",
  "contributing_ECMs": ["ECM 1 name", "ECM 2 name", "ECM 3 name"],
  "benefits": {"energy savings increase": 0, "cost reduction": 0.15, "source": {
    "notes": "Explanatory text related to source data and/or values given.",
    "source_data": [{
      "title": "Reference Title",
      "author": "Author Name(s)",
      "organization": "Organization Name",
      "year": "2016",
      "pages": null,
      "URL": "http://buildings.energy.gov/"}}]}},
{"name": "Second package name",
  "contributing_ECMs": ["ECM 4 name", "ECM 1 name"],
  "benefits": {"energy savings increase": 0.03, "cost reduction": 0.18, "source": {
    "notes": "Explanatory text regarding both energy savings and cost reduction values_
→given.",
    "source_data": [{
      "title": "Reference Title",
      "author": "Author Name(s)",
      "organization": "Organization Name",
      "year": "2016",
      "pages": "238-239",
      "URL": "http://buildings.energy.gov/"}}]}},
{"name": "Third package name",
  "contributing_ECMs": ["ECM 5 name", "ECM 3 name", "ECM 6 name", "ECM 2 name"],
  "benefits": {"energy savings increase": 0.2, "cost reduction": 0, "source": {
    "notes": "Explanatory text related to source data and/or values given.",
    "source_data": [{
      "title": "Reference Title",
      "author": "Author Name(s)",
      "organization": "Organization Name",
      "year": "2016",
      "pages": "82",
      "URL": "http://buildings.energy.gov/"}}]}]}
```

1.5.2 Tutorial 2: Running with project configuration files

Arguments to the ecm_prep.py and run.py scripts can be defined using a .yaml configuration file. Arguments in a project definition configuration file map to the command-line arguments described in the “Additional options” sections in *Tutorial 3* (ecm_prep.py) and *Tutorial 5* (run.py), but enable a consistent and reusable approach to running Scout.

Writing configuration files

To get started writing a configuration file, users can reference the [sample configuration file](#) found on the Scout repository, which serves as a valid configuration file pre-filled with default values. Update any relevant fields required to configure a custom scenario; any unchanged arguments can be left as is or deleted. Shown below is an easily readable version of the Scout yaml schema; this reflects information shown when running `ecm_prep.py` and `run.py` with `--help`, but also shows the expected structure of an input yaml file.

```
description: (string) A description of the scenario configuration.
  Default null
ecm_prep:
  add_typ_eff: (boolean) If true, enable automatic addition of
    ECMs with typical (business-as-usual) efficiency to compete
    with user-defined ECMs. Default False
  adopt_scn_restrict: (string) Specify single desired adoption
    scenario, otherwise both scenarios will be used. Allowed values
    are {Max adoption potential, Technical potential, null}. Default
    null
  alt_ref_carb: (string) Specify alternate reference grid scenario
    for baseline electricity emissions intensities (rather than
    AEO). 'MidCase' refers to the Standard Scenarios Mid Case
    and 'DECARB-bau' refers to the ReEDS "DECARB" business-as-usual
    scenario. Allowed values are {DECARB-bau, MidCase, null}.
    Default null
  alt_regions: (string) Specify an alternative region breakdown.
    Allowed values are {AIA, EMM, State}. Default EMM
  captured_energy: (boolean) If true, enable captured energy calculation.
    Default False
  detail_brkout: (array) List of options by which to breakout
    results. The 'fuel types' option is only valid if the split_fuel
    argument is set to false. The 'all' option selects all three
    breakout categories. The 'codes/bps' option sets the minimum
    building type/vintage breakouts needed to assess codes and
    performance standards. Allowed values are 0 or more of {all,
    buildings, codes/bps, fuel types, regions}. Default []
  ecm_directory: (string) Directory containing ECM definitions
    and savings shapes (if applicable). The directory path can
    be absolute or relative to either the configuration file (.yaml)
    or the command line path, depending on where the argument
    is assigned. If not provided, ./ecm_definitions will be used.
    Default null
  ecm_field_updates: (object) Key-value pairs to update fields
    across all ECM definitions. Any number of ECM fields can be
    updated with additional keys. Keys should correspond to ECM
    definition fields. Default null
  ecm_files: (array) Specify a subset of ECM definitions in the
    ECM directory. Values must contain exact matches to the ECM
    definition file names in 'ecm_directory', excluding the file
    extension. If argument is null, then all files within 'ecm_directory'
    will be prepared, subject to the 'ecm_files_regex' argument,
    if applicable. Default null
  ecm_files_regex: (array) Specify a subset of ECM definitions
    in the ECM directory using a regular expression. Each value
    will be matched to file names in 'ecm_directory'. Default
```

(continues on next page)

(continued from previous page)

`[]`

ecm_packages: (array) Specify ECM packages; values must correspond with package_ecms.json package names in the ECM directory. If no list or an empty list is provided, no packages will be used. Include an asterisk ("*") in the list to prepare all packages present in package_ecms.json. Contributing ECMs for specified packages will automatically be prepared regardless of their presence in the ``ecm_directory``, ``ecm_files``, and/or ``ecm_files_regex`` arguments. Default `[]`

elec_upgrade_costs: (string) Determine how to assign (or not) electrical infrastructure upgrade costs. Allowed values are {all, ignore, shares, null}. Default shares

exog_hp_rates:

exog_hp_rate_scenario: (string) Conversion scenario to use for exogenous rates of conversion to HPs/electric equipment. May be drawn from legacy Guidehouse ('gh-') conversion scenarios or scenario(s) that were produced via endogenously running measure competition and pulling out conversion rates from results. Allowed values are {bss-brk, gh-aggressive, gh-conservative, gh-most aggressive, gh-optimistic, null}. Default null

switch_all_retrofit_hp: (boolean) Only relevant to Guidehouse scenarios. If true, assume all retrofits convert to heat pumps/electric alternative, otherwise, retrofits are subject to the same external heat pump conversion rates assumed for new/replacement decisions. Only applicable if ``exog_hp_rate_scenario`` and ``retrofit_type`` are defined. Default False

floor_start: (integer) Apply an elevated performance floor starting in the year specified. The desired future year must be given in the format YYYY. Default null

fugitive_emissions: (array) Array enabling fugitive emissions; array may include one of the three supply chain methane leakage rate options and one of the three refrigerant options. Allowed values are 0 or more of {low-gwp refrigerant, methane-high, methane-low, methane-mid, typical refrigerant, typical refrigerant no phaseout}. Default `[]`

grid_decarb:

grid_assessment_timing: (string, required if the grid_decarb key is present) When to assess avoided emissions and costs from non-fuel switching measures relative to the additional grid decarbonization. Only applicable if ``grid_decarb_level`` is provided. Allowed values are {after, before, null}. Default null

grid_decarb_level: (string, required if the grid_decarb key is present) Enable grid decarbonization - ``95by2050`` represents a 95 percent reduction in grid emissions from 2005 levels by 2050 and ``100by2035`` represents 100 percent grid decarbonization by 2035. 'DECARB' options were produced as part of EERE's DECARB initiative; ``DECARB-mid`` represents 97 percent grid decarbonization by 2050, and ``DECARB-high`` represents 100 percent grid decarbonization by 2035. Only applicable if ``grid_assessment_timing`` is also provided. Allowed values are {100by2035, 95by2050, DECARB-high, DECARB-mid, null}.

(continues on next page)

(continued from previous page)

Default null

health_costs: (boolean) If true, enable health costs. Requires alt_regions to be set to `EMM`. Default False

incentives:

- incentive_levels:** (['string']) Level of federal and state incentives (e.g., tax credits, rebates) for measure adoption. The aeo option reverts to incentive levels in the AEO reference case forecast. Allowed values are {aeo, aggressive, aggressive state, optimistic, reference}. Default aeo
- incentive_restrictions:** (string) Restrictions to apply to evaluation of incentives. Setting to all removes all incentives, including those in the reference case, while null imposes no additional restrictions. Allowed values are {all, no electric, no federal, no federal electric, no federal fossil, no fossil, no ira, no sub-fed fossil, no sub-federal, no sub-federal electric, null}. Default null

no_eff_capt: (boolean) If true, suppress reporting of ECM-captured efficient energy use. Default False

no_lnkdcosts:

- no_lnkdcosts_op:** (boolean) For measures that span heating or cooling and other end uses, count operating costs for only the anchor end use (rather than all end uses) in representing competition w/ other measures (default anchor is heating). Default False
- no_lnkdcosts_stk:** (string) For measures that span heating or cooling and other end uses, count stock costs for only the anchor end use (rather than all end uses) in representing competition w/ other measures (default anchor is heating). Allowed values are {in_adopt_and_report, in_adopt_only, null}. Default null

no_scnd_lgt: (boolean) If true, disable the calculation of secondary heating and cooling energy effects from changes in lighting efficacy. Default False

pkg_env_costs: (string) Define what measure data should be written out for inclusion in measure competition. `include HVAC` will prepare HVAC-only versions of all HVAC/envelope packages for measure competition, `exclude HVAC` will exclude these HVAC-only measures from measure competition. Allowed values are {exclude HVAC, include HVAC, null}. Default null

pkg_env_sep: (boolean) If true, enable output of separate envelope ECM impacts when both HVAC and envelope ECMs are combined in one or more measure packages. Default False

price_sensitivity: (string) Use supply-side scenarios with relatively high electricity prices/low natural gas prices, or vice versa. Will not be assessed if grid_decarb_level is non-null. Allowed values are {highelec_lowfossil, lowelec_highfossil, null}. Default null

report_stk:

- cap_invest:** (boolean) Report baseline and measure unit capital costs, including summaries across segments and full breakouts. Default False
- num_units:** (boolean) Report number of baseline and measure

(continues on next page)

(continued from previous page)

units, including summaries across segments and full breakouts.
Default False

retrofits:

- retrofit_mult_year:** (integer) The year by which the retrofit multiplier is achieved (only for increasing retrofit_type).
Default null
- retrofit_multiplier:** (number) Factor by which early retrofit rates are multiplied (only for increasing retrofit_type).
Default null
- retrofit_type:** (string, required if the retrofits key is present)
Specify early retrofit rates. `constant` assumes component-based early retrofit rates that do not change over time, `increasing` assumes rates that increase over time. Allowed values are {constant, increasing, null}. Default null

rp_persist: (boolean) If true, enable relative performance persistence.
Default False

sect_shapes: (boolean) If true, enable calculation of sector-level electricity savings shapes. Default False

site_energy: (boolean) If true, enable site energy calculation.
Default False

split_fuel: (boolean) If true, split out ECM results reporting by fuel type. Default False

sub_fed_policy:

- bps:** (string) State or local Building Performance Standards (BPS) including energy use index targets and/or onsite emissions reduction requirements, which apply to existing buildings. Null removes this driver entirely. Allowed values are {aggressive, optimistic, reference, null}. Default null
- bps_comply_com:** (number) Commercial compliance fraction to assume for BPS scenario implementation. Default 1
- bps_comply_res:** (number) Residential compliance fraction to assume for BPS scenario implementation. Default 1
- code_comply_com:** (number) Commercial compliance fraction to assume for code scenario implementation. Default 1
- code_comply_res:** (number) Residential compliance fraction to assume for code scenario implementation. Default 1
- codes:** (string) State or local new building codes w/ additional energy index improvements over model code and/or onsite emissions reduction requirements/incentives. Null removes this driver entirely. Allowed values are {aggressive, optimistic, reference, null}. Default null
- low_volume_rate:** (string) Reduce volumetric electric rates, possibly with concurrent increase in fixed charges. Reference case will represent currently-enacted rates of this type. Null removes this driver entirely. Allowed values are {aggressive, optimistic, reference, null}. Default null
- state_appl_regs:** (string) State or local emissions-related regulations that remove certain appliance choices from the market. Null removes this driver entirely. Allowed values are {aggressive, optimistic, reference, null}. Default null

tsv_metrics:

- tsv_average_days:** (string) The day type to average over. Required

(continues on next page)

(continued from previous page)

if `tsv_type` is ``energy`` or `tsv_power_agg` is ``average``. Allowed values are {all, weekdays, weekends, null}. Default null

tsv_daily_hr_restrict: (string, required if the `tsv_metrics` key is present) The daily hour range to restrict time-sensitive metrics, where ``all`` is all hours, ``peak`` is peak demand period hours, and ``low`` is low demand period hours. Required if running `tsv_metrics`. Allowed values are {all, low, peak, null}. Default null

tsv_energy_agg: (string) Define how the the `tsv_daily_hr_restrict` hours are aggregated, required when `tsv_type` is ``energy``. Allowed values are {average, sum, null}. Default null

tsv_power_agg: (string) Define how the the `tsv_daily_hr_restrict` hours are aggregated, required when `tsv_type` is ``power``. Allowed values are {average, peak, null}. Default null

tsv_season: (string) Season of focus for time-sensitive metrics. Allowed values are {intermediate, summer, winter, null}. Default null

tsv_sys_shape_case: (string) The basis for determining hours for peak or low demand. Required if `tsv_daily_hr_restrict` is ``peak`` or ``low``. Allowed values are {net renewable high renewables, net renewable reference, total high renewables, total reference, null}. Default null

tsv_type: (string, required if the `tsv_metrics` key is present) Time-sensitive metric desired, where ``energy`` is the change in energy and ``power`` is the change in power (e.g., single hour GW). Required if running `tsv_metrics`. Allowed values are {energy, power, null}. Default null

verbose: (boolean) If true, enable verbose mode. Default False

run:

high_res_comp: (boolean) If true, resolve competition cost data to each market microsegment (rather than averaging across all applicable markets). Default False

mkt_fracs: (boolean) If true, flag market penetration outputs. Default False

no_comp: (boolean) If true, suppress measure competition. Default False

report_cfs: (boolean) If true, report competition adjustment fractions. Default False

report_custom_yrs: (array) Enter a custom list of years to use for reporting. When left blank, all years are reported. Default []

results_directory: (string) Directory in which to output results. The directory path can be absolute or relative to either the configuration file (.yaml) or the command line path, depending on where the argument is assigned. If not provided, ./results will be used. Default null

trim_vars: (boolean) If true, a reduced set of only essential variables is reported. Default False

verbose: (boolean) If true, print all warnings to stdout. Default False

write_elec_conv_fracs: (boolean) If true, writes out rates of either fuel or tech. switching to electric equipment. Default

(continues on next page)

(continued from previous page)

False

Running with a single configuration file

To run `ecm_prep.py` or `run.py` with a yaml configuration file you can run one or both of the following:

```
python scout/ecm_prep.py -y <my_project.yml>
python scout/run.py -y <my_project.yml>
```

Scout will parse the .yaml file and write arguments for each script, provided there is a corresponding `ecm_prep` and/or `run` key specifying `ecm_prep.py` or `run.py` arguments, respectively.

Note

If other command-line arguments are included (i.e., those other than `-y`), then they will take precedence over the yaml file if there is overlap between the two.

Running with multiple configuration files

Batches of configuration files can also be run using the `run_batch.py` module. This automatically runs both `ecm_prep.py` and `run.py` for a set of configuration files stored in a common directory. To utilize this feature, first generate any number of configuration files (i.e., scenarios) and store them in a common directory of any name. To run the scenarios, enter the following command line argument, where `<config_directory>` refers to the directory containing the set configuration files:

Windows

```
python scout\run_batch.py --batch <config_directory>
```

Mac

```
python3 scout/run_batch.py --batch <config_directory>
```

To minimize redundant data processing, configuration files sharing identical `ecm_prep` arguments are consolidated into groups. Artifacts produced by `ecm_prep.py` are then stored in directories labeled as `generated/batch_run<n>`, where `<n>` increments with each group. Additionally, each configuration file is copied into its respective group directory.

Final results for each scenario are written to the directory specified in the `run.py --results_directory` argument (see [Tutorial 5](#)), or are defaulted to a directory matching the configuration file name within the `./results` folder. Because this module runs both core Scout modules (`ecm_prep.py` and `run.py`), Tutorials 3-6 are not relevant if running with `run_batch.py`.

1.5.3 Tutorial 3: Preparing ECMs for analysis

The Scout analysis is divided into two steps, each with corresponding Python modules. In the first of these steps, discussed in this tutorial, the ECMs are pre-processed by retrieving the applicable baseline energy, CO₂, and cost data from the input files (located in the `./scout/supporting_data/stock_energy_tech_data` directory) and calculating the uncompleted efficient energy, CO₂, and cost values. This pre-processing step ensures that the computationally intensive process of parsing the input files to retrieve and calculate the relevant data is only performed once for each new or edited ECM.

Each new ECM that is written following the formatting and structure guidelines covered in [Tutorial 1](#) should be saved in a separate JSON file with a brief but descriptive file name and placed in a common directory. This directory can be specified with the `ecm_directory` argument described in [Specify ECM files and packages](#), or it will be defaulted to

`./ecm_definitions` . If any changes to the package ECMs are desired, incorporating either or both new and existing ECMs, follow the instructions in the [package ECMs](#) section to specify these packages. The pre-processing script can be run once these updates are complete.

To run the pre-processing script `ecm_prep.py`, open a Terminal window (Mac) or command prompt (Windows), navigate to the Scout project directory (shown with the example location `./Documents/projects/scout-run_scheme`), and run the script.

Windows

```
cd Documents\projects\scout-run_scheme
python scout\ecm_prep.py
```

Mac

```
cd Documents/projects/scout-run_scheme
python3 scout/ecm_prep.py
```

As each ECM is processed by `ecm_prep.py`, the text “Updating ECM” and the ECM name are printed to the command window, followed by text indicating whether the ECM has been updated successfully. There may be some additional text printed to indicate whether the installed cost units in the ECM definition were converted to match the desired cost units for the analysis. If any exceptions (errors) occur, the module will stop running and the exception will be printed to the command window with some additional information to indicate where the exception occurred within `ecm_prep.py`. The error message printed should provide some indication of where the error occurred and in what ECM. This information can be used to narrow the troubleshooting effort.

If `ecm_prep.py` runs successfully, a message with the total runtime will be printed to the console window. The names of the ECMs updated will be added to `./generated/run_setup.json`, a file that indicates which ECMs should be included in [the analysis](#). The total baseline and efficient energy, CO₂, and cost data for those ECMs that were just added or revised are added to the `./generated/ecm_competition_data` folder, where there appear separate compressed files for each ECM. High-level summary data for all prepared ECMs are added to the `ecm_prep.json` file in the `./generated` folder. These files are then used by the ECM competition routine, outlined in [Tutorial 5](#).

Tip

The format of `ecm_prep.json` is a list of dictionaries, with each dictionary including one ECM’s high-level summary data. Use the name key in these ECM summary data dictionaries to find information for a particular ECM of interest in this file.

If exceptions are generated, the text that appears in the command window should indicate the general location or nature of the error. Common causes of errors include extraneous commas at the end of lists, typos in or completely missing keys within an ECM definition, invalid values (for valid keys) in the specification of the applicable baseline market, and units for the installed cost or energy efficiency that do not match the baseline cost and efficiency data in the ECM.

Additional preparation options

Users may include a range of additional options alongside the `ecm_prep.py` command that modify default ECM preparation settings.

Windows

```
python scout\ecm_prep.py <additional option 1> <additional option 2> ... <additional_
↵option N>
```

Mac

```
python3 scout/ecm_prep.py <additional option 1> <additional option 2> ... <additional_
↳option N>
```

The additional ECM preparation options are described further here.

Configuration file

`--yaml` specifies the filepath to a yaml configuration file, as described in the [Tutorial 2](#). This file provides an alternative way of defining the command line arguments below. Additional arguments passed via the command line will take precedence over those defined in the yaml file.

Specify ECM files and packages

`--ecm_directory` defines the directory that stores all ECM definition files, savings shapes, and the `package_ecms.json` file. If not provided, `./ecm_definitions` will be used.

Note

The default path, `./ecm_definitions`, is referenced throughout the documentation, but this path can be changed with the use of this argument.

`--ecm_files` selects a subset of ECM definitions within the ECM directory. Values must contain exact matches to the ECM definition file names in `--ecm_directory`, excluding the file extension.

`--ecm_files_regex` selects a subset of ECM definitions within the ECM directory using a list of regular expressions. Each value will be matched to file names in `--ecm_directory`.

`--ecm_packages` selects a subset of ECM packages; values must correspond with `package_ecms.json` package names in the ECM directory. If no list is provided, all packages will be used so long as their ECM definitions are present; if an empty list is provided, no packages will be used. Packages that require ECM definitions not specified through `--ecm_directory`, `--ecm_files`, and/or `--ecm_files_regex` will be skipped and a warning will be issued.

Alternate regions

`--alt_regions` allows the user to set the regional breakout of baseline data and ECM results (see [Climate zone](#)). Valid options include one of “EMM” (default), “State”, or “AIA”.

Note

Currently, three regional breakouts are supported: the U.S. Electricity Information Administration (EIA) Electricity Market Module (EMM) regions, AIA climate regions, and the contiguous U.S. states. See the [Climate zone](#) section for additional details.

Site energy

`--site_energy` prepares ECM markets and impacts in terms of site energy use, rather than in terms of primary (or source) energy use as in the default ECM preparation.

Restricted adoption scenarios

`--adopt_scn_restrict` limits ECM preparation and analysis to one of the two default adoption scenarios (see *ECMs are adopted one of two ways*). Valid options include either “Max adoption potential” or “Technical potential”.

Detailed results breakouts

`--detail_brkout` reports regional, building type, and/or fuel type breakouts of results at the highest possible resolution. Valid options include at least one of “regions”, “buildings”, “fuel types”, and “all”. If “regions” is included, then `--alt_regions` must be set to “EMM”. If “fuel types” is included, then `--split_fuel` must be false.

Note

Default regional breakouts depend on the *region selection* for the current run. An *AIA* region selection does not have a more detailed breakout option. An *EMM* region selection defaults to reporting breakouts for a higher-level aggregation of those 25 regions into 11 broader regions that are similar to the 2019 EPA AVERT regions but separate the Great Basin from the Northwest region; the detailed breakout option resolves results by all 25 EMM regions. Finally, a *U.S. state* region selection defaults to reporting breakouts by the 9 U.S. Census Divisions; the detailed breakout option resolves results by all contiguous U.S. states plus the District of Columbia.

Default building type breakouts are resolved by residential vs. commercial and by vintage (new — constructed at or after the start of the modeling time horizon — vs. existing). Detailed building type breakouts further resolve the building types into 2 residential and 8 commercial types, while dropping the split by vintage (single family/mobile homes and multi family homes for residential; hospitals, large offices, small/medium offices, retail, hospitality, education, assembly/other, and warehouses for commercial).

High electric grid decarbonization

`--grid_decarb_level` selects versions of annual and hourly electricity emissions and price inputs that are consistent with a more aggressive decarbonization pathway for the electric grid than is assumed in the default *AEO Reference Case*. When this argument is passed, the user must specify either “95by2050”, “100by2035”, “DECARB-mid”, or “DECARB-high” to define the additional grid decarbonization scenario. “95by2050” represents a 95 percent reduction in grid emissions from 2005 levels by 2050 and “100by2035” represents 100 percent grid decarbonization by 2035. ‘DECARB’ options were produced as part of EERE’s DECARB initiative; “DECARB-mid” represents 97 percent grid decarbonization by 2050, and “DECARB-high” represents 100 percent grid decarbonization by 2035. This argument is only applicable if `--grid_assessment_timing` is also provided.

`--grid_assessment_timing` selects whether avoided emissions and costs from non-fuel switching measures should be assessed *before* or *after* accounting for additional grid decarbonization beyond the Reference Case, by specifying either “before” or “after” with this argument. Avoided emissions and costs for fuel switching measures will always be assessed *after* accounting for additional grid decarbonization beyond the Reference Case. This argument is only applicable if `--grid_decarb_level` is also provided.

Note

Annual emissions intensities for the more aggressive grid decarbonization scenarios are drawn from the *NREL Cambium scenarios* and the *Scout DECARB scenarios* and are found in `./scout/supporting_data/convert_data`. Annual electricity price data (also found in `./scout/supporting_data/convert_data`) and hourly electricity emissions and price data for the more aggressive grid decarbonization scenarios (found in `./scout/supporting_data/tsv_data`) are drawn from different sources — the *EIA Annual Energy Outlook Low Zero-carbon Technology Cost* for the annual electricity price data, and the *NREL Cambium Low Renewable Energy Cost Scenario* for the hourly data.

Note

Currently the `--grid_decarb` option is not supported for state regions; if state regions are selected alongside the `--grid_decarb` option, the code will automatically switch the run to EMM regions while warning the user.

Exogenous heat pump switching rates

`--exog_hp_rate_scenario` imposes externally determined rates of technology and/or fuel switching from fossil- or resistance-based equipment to heat pump technologies, with the default rates available in `./scout/supporting_data/convert_data/hp_convert_rates`. When this option is provided, users must select one of four scenarios of switching rates: “conservative”, “optimistic”, “aggressive”, or “most aggressive”. These scenarios were developed by Guidehouse as benchmarks for the U.S Department of Energy’s [E3HP Initiative](#).

Note

In the absence of the `--exog_hp_rate_scenario` option, rates of switching to heat pump measures are determined based on a tradeoff of the capital and operating costs of the candidate heat pump measures against those of competing measures in the analysis, as described in [Competing ECMs and updating savings](#).

Note

Currently the `--exog_hp_rate_scenario` option is not supported for AIA climate regions; if AIA climate regions are selected alongside the `--exog_hp_rate_scenario` option, the code will automatically switch the run to EMM regions while warning the user.

`--switch_all_retrofit_hp` selects whether the exogenous rates should be applied to early retrofit decisions (as well as to decisions regarding regular replacements and new construction) or if all early retrofit decisions should be assumed to switch to the candidate heat pump technology. Note that while the exogenous rates were developed to describe rates of switching of heating and water heating technologies to heat pumps, rates of natural gas heating conversions are also applied to the cooking end use. This argument is only applicable if `--exog_hp_rate_scenario` is specified.

Assessment of fugitive emissions

`--fugitive_emissions` enables assessment of CO₂-equivalent emissions from two fugitive sources: 1) increased emissions from leakage of equipment refrigerants (e.g., for HVAC and refrigeration equipment), and 2) avoided emissions from reducing natural gas consumption and its fugitive emissions from leakage throughout the natural gas supply chain. Supplementary data and reference information for both of these sources are available in `./scout/supporting_data/convert_data/fugitive_emissions_convert.json`. When this option is selected, the user must provide at least one of “methane-low”, “methane-mid”, “methane-high”, “low-gwp refrigerant”, “typical refrigerant”, and “typical refrigerant no phaseout”. Valid options include one option, a combination of one of the three “methane*” and one of the three “*refrigerant*” options. When including more than one, Scout will assess fugitive emissions for the sources together. For fugitive emissions from methane leakage, the user must specify whether lower bound methane leakage rates (“methane-low”), mid-range methane leakage rates (“methane-mid”), or upper bound methane leakage rates (“methane-high”) are desired. For fugitive emissions from equipment refrigerant leakage, the user will specify whether to assume that measures use market-available refrigerants and that those refrigerants phase out according to U.S. EPA’s phase-out rules under the [Significant New Alternatives Policy \(SNAP\)](#) (“typical refrigerant”), that measures use market-available refrigerants with no phase-out requirements (“typical refrigerant no phaseout”) or to assume that measures use low-GWP refrigerants (“low-gwp refrigerant”).

Note

Currently the `--fugitive_emissions` option is not supported for AIA climate regions; if AIA climate regions are selected alongside the `--fugitive_emissions` option, the code will automatically switch the run to EMM regions while warning the user.

Persistent relative performance

`--rp_persist` calculates the market entry energy performance and installed cost of each ECM being prepared relative to its comparable baseline technology, and maintains this relative energy performance and cost across the full modeling time horizon. For example, if ECMs are 10% more efficient and 10% higher cost than comparable baseline technologies at market entry, they will still be 10% more efficient and higher cost than comparable baseline technologies by the end of the modeling time horizon.

Fuel splits

`--split_fuel` prepares fuel type breakouts for measure results that are reported to the file `./generated/ecm_prep.json` under the `mseg_out_break` dict key and carried through to the measure results file `./results/ecm_results.json` that is generated by `run.py`. Fuel type breakouts will be reported under `Electric` and `Non-Electric` dict keys; the splits are nested under higher-level breakouts of the results by region, building type/vintage, and end use. Fuel splits are only reported out for the heating, water heating, and cooking end uses.

Add Reference Case measures

`--add_typ_eff` automatically prepares [AEO Reference Case](#) copies of any measures that the user has flagged as requiring a competing reference case analogue (via the `ref_analogue` attribute). The Reference Case measures feature no incremental cost, performance, or lifetime differences from the baseline technologies they apply to (determined via the measures' `technology` attribute). They are otherwise identical to the original measure in their baseline market characteristics. Data for these measures are prepared and reported just like any other measure, such that they will factor into any measure competition simulated down the line in the `run.py` routine.

Note

All Reference Case analogue measures will include the string “(Ref. Analogue)” in their reported name, so that these measures are readily flagged in data post-processing.

Raise technology performance floor

`--floor_start [year]` sets a year by which any measures at the ENERGY STAR, IECC, and/or 90.1 performance levels in the analysis (as identified by those measures' `name` attribute) represent the minimum performance level for market-available technologies. Beginning in that year, any Reference Case technologies in the analysis (specified via the `--add_typ_eff` option above) will exit the market and will no longer factor into measure competition. If a user has not represented any Reference Case technologies in the measure set (e.g., they have not specified the `--add_typ_eff` option), the year specified alongside `floor_start` will override the `market_entry_year` attribute for all measures in the analysis and no measure will enter the market before that year.

Note

The lowest-performing measures in a Scout analysis act as a performance “floor” for the building technology options that are market-available in a given year and thus operate akin to a minimum energy performance code or standard. The `--floor_start` option may be useful in exploring the effects of implementing a global minimum performance

level that is consistent with current ENERGY STAR/IECC/90.1 specifications by different years in the modeling time horizon.

Specify early retrofit rates

--retrofit_type assumes that a certain portion of technologies are replaced before the end of their useful life each year at a component-specific rate, on top of the portion that is regularly replaced at end-of-life. When this option is passed, the user must specify one of “constant” or “increasing”, which defines whether the early retrofit rates should remain constant over time or escalated to achieve a certain multiplier by a certain year (e.g., 4X the starting rates by 2035), respectively. If applying a multiplier (“increasing”), then --retrofit_multiplier and --retrofit_mult_year must also be specified. Component-specific rate assumptions are reported in [Table 4](#).

--retrofit_multiplier designates the factor by which early retrofit rates are multiplied. Only applicable if --retrofit_type is specified.

--retrofit_mult_year designates the year by which the retrofit multiplier is achieved. Only applicable if --retrofit_type is specified.

Table 4: Assumed values and sources for component-specific early retrofit rates.⁸

Building Type	Data Source	Component Retrofitted (Year Range)	Annual Rate (%)
Commercial	CBECS 2012	Lighting (2000-2008)	1.5
		HVAC (1990-2008)	0.9
		Roof (1990-2008)	0.6
		Windows (1990-2008)	0.3
		Insulation (1990-2008)	0.3
	Use com. HVAC	Water Heating	0.9
Residential	N/A	All Other	0
	AHS 2019	HVAC (1990-2008)	0.5
		Roof (1990-2008)	0.27
		Windows (1990-2008)	0.23
		Insulation (1990-2008)	0.06
	Use res. HVAC	Water Heating	0.5
	Use com. lgt.	Lighting	1.5
	N/A	All Other	0

Isolate W/E impacts in ECM packages

--pkg_env_sep automatically generates and reports data for windows and envelope (W/E)-only versions of any *ECM packages* in the analysis that couple HVAC equipment improvements with W/E upgrades (HVAC + W/E). Data for these W/E-only counterfactual packages are reported to a separate ecm_prep file, ./generated/ecm_prep_env_cf.json .

Note

All counterfactual packages in ecm_prep_env_cf.json will share the names of the original packages that they are derived from, but with the string “(CF)” appended.

⁸ To produce the rate estimates in the table, we focus on the proportion of buildings in the data that report retrofitting a given technology before the end of its expected useful lifetime. For example, for commercial HVAC equipment, we find the total number of buildings constructed between 1990 and 2008 that report an HVAC renovation during that period, under the assumption that HVAC equipment typically functions for 20 years and thus would not be regularly replaced until 2010 at the earliest (1990 — the earliest construction year in the building sample — plus 20 years). We divide that number by the total number of buildings constructed in that time period, and annualize by dividing the result by 18 years (2008-1990).

 Tip

W/E-only counterfactual package data in `ecm_prep_env_cf.json` may be compared with data for the parallel HVAC + W/E package in `ecm_prep.json` to isolate the impacts of the W/E portion of the package on overall package results.

Reflect W/E costs in ECM packages

`--pkg_env_costs` reflects the installed cost of W/E technologies that are included in HVAC + W/E *ECM packages* in the overall installed costs for the package. The user must specify either “include HVAC” or “exclude HVAC” as the value for this argument.

 Note

By default, W/E costs are excluded from the overall costs of an HVAC + W/E package to harmonize the handling of costs in such packages with the approach of Scout’s technology choice models, which are drawn from EIA National Energy Modeling System (NEMS) data on HVAC equipment costs and sales only.

Alternative emissions intensities

`--alt_ref_carb` sets the baseline electricity emissions intensities to be consistent with the [Standard Scenarios Mid Case](#) (rather than AEO).

Time sensitive valuation metrics

The following time sensitive valuation metrics are used to assess and report out ECM impacts on electric load during pre-defined sub-annual time slices, rather than impacts on annual energy use as in the default ECM preparation. Time slice settings are based on 2006 as the [reference year](#) for the purpose of defining the days of the week and number of days in the year.

`--tsv_type` selects the reported metric to represent either change in energy use across multiple hours (e.g., kWh, GWh, TWh) or change in power per hour (e.g., kW, GW, TW). Valid options include “energy” or “power”.

`--tsv_daily_hr_restrict` specifies the daily hour range to restrict tsv metrics to. The time slice can include all 24 hours of a day or be set to specific a daily period of peak demand on the electric grid (e.g., 4–8 PM) or low demand on the electric grid (e.g., 12–4 AM). Valid options include “all”, “peak”, or “low”.

`--tsv_sys_shape_case` specifies the basis for determining hour range. Periods of peak and low demand are determined using system-level load profiles for a representative set of [EMM regions](#). These profiles and associated periods may be based on *total* system demand, or total system demand *net* renewable energy generation. Furthermore, the system profiles may be based on either the [AEO Reference Case](#) or the [AEO Low Oil and Gas Supply](#) (e.g., higher renewable penetration) side case assumptions. Valid options include “total reference”, “total high renewables”, “net renewable reference”, or “net renewable high renewables”. This argument is only applicable if `--tsv_daily_hr_restrict` is set to “peak” or “low”.⁹

`--tsv_season` limits the analysis to one of three seasons: summer (Jun–Sep), winter (Dec–Mar), or intermediate (Oct–Nov, Apr–May). Valid options include “summer”, “winter”, or “intermediate”.

`--tsv_energy_agg` defines how metrics are aggregated when `--tsv_type` is set to “energy”. This argument will specify a sum or average across the hours specified in `--tsv_daily_hr_restrict`. Valid options include “sum” or

⁹ Total and net peak and low demand hour ranges by season, EMM region, and projection year are summarized in the files `./scout/supporting_data/tsv_data/tsv_hrs_net_ref.csv`, `./scout/supporting_data/tsv_data/tsv_hrs_net_hr.csv`, `./scout/supporting_data/tsv_data/tsv_hrs_tot_ref.csv`, and `./scout/supporting_data/tsv_data/tsv_hrs_tot_hr.csv`. The default periods assumed when a user adds the `--tsv_metrics` option reflect the projection year 2050 and a representative subset of system load shapes from 16 EMM regions: BASN, CANO, CASO, FRCC, ISNE, MISS, MISE, MISW, NWPP, PJME, PJMW, RMRG, SRCE, SRSE, SRSG, TRE. For a graphical example of the *net* system load shapes and peak/low demand period definitions used to support the `--tsv_metrics` option, refer to [this plot](#).

“average”.

`--tsv_power_agg` defines how metrics are aggregated when `--tsv_type` is set to “power”. This argument will specify a maximum or average across the hours specified in `--tsv_daily_hr_restrict`. Valid options include “peak” or “average”.

`--tsv_average_days` defines the day type to average over. Valid options include “all”, “weekdays”, or “weekends”. This argument is only applicable if `--tsv_type` is “energy” or `tsv_power_agg` is “average”.

Note

When the `--tsv_metrics` option is used, all data prepared for the ECM and written out to `./generated/ecm_competition_data` and `./generated/ecm_prep.json` will reflect the specific time slice of interest, rather than the default annual outcomes.

Note

Data needed to support evaluation of TSV metrics are broken out by EMM region; when using the `--tsv_metrics` option, EMM should be selected as the regional breakout when running `ecm_prep.py`. If regions are not set to EMM in this case, the code will do so automatically while warning the user.

Sector-level hourly energy loads

`--sect_shapes` modifies the results output to `./generated/ecm_prep.json` to include, for each ECM, the hourly energy use (in MMBtu) attributable to the portion of the building stock the ECM applies to in a given adoption scenario, EMM region, and projection year, both with and without the measure applied. These hourly energy loads are reported for all 8760 hours of a year that corresponds to a [reference year](#).

Note

Sector-level 8760 load data for an ECM are written to the “sector_shapes” key within the given ECM’s dictionary of summary data in `./generated/ecm_prep.json`. The 8760 load data are nested in another dictionary under the “sector_shapes” key according to the following key hierarchy: adoption scenario (“Technical potential” or “Max adoption potential”) -> EMM region (see [EIA Electricity Market Module \(EMM\) regions](#) for names) -> summary projection year (“2020”, “2030”, “2040” or “2050”) -> efficiency scenario (“baseline” or “efficient”). The terminal values at the end of each key chain will be a list with 8760 values.

Public health benefits

`--health_costs` adds low and high estimates of the public health cost benefits of avoided fossil electricity generation from the deployment of each ECM being prepared. The low and high public health cost benefits estimates are drawn from the “Uniform EE - low estimate, 7% discount” and “Uniform EE - high estimate, 3% discount” cases in the [U.S. Environmental Protection Agency \(EPA\) report “Public Health Benefits per kWh of Energy Efficiency and Renewable Energy in the United States: a Technical Report”](#).¹⁰¹¹

¹⁰ With this option, low/high estimates of public health benefits are added directly to electricity costs, yielding greater savings for ECMs that are able to reduce electricity use.

¹¹ The EPA report also includes low and high estimates of the public health benefits of avoided electricity generation from energy efficiency during the peak hours of 12-6 PM. While these estimates are ultimately very similar to the “Uniform EE” estimates and not included in Scout’s health cost adders, they are summarized by region alongside the “Uniform EE” estimates in the file `./scout/supporting_data/convert_data/epa_costs.csv`.

Note

Public health cost adders are broken out by EMM region; when using the `--health_costs` option, EMM should be selected as regional breakout when running `ecm_prep.py`. If regions are not set to EMM in this case, the code will do so automatically while warning the user.

Note

When ECMs are prepared with the public health cost adder, three versions of the ECM will be produced: 1) the ECM prepared according to defaults, *without* health cost adders, 2) a version of the the ECM with a low public health cost adder `<ECM Name> - PHC-EE (low)`, and 3) a version of the ECM with a high public health cost adder `<ECM Name> - PHC-EE (high)`. Since the EPA [report](#) estimates public health benefits based on the current fossil fuel generation mix, **users are advised against retaining any results for ECMs prepared with public health cost adders beyond the year 2025.**

Public health cost adders only apply to the `electricity` *fuel_type*. If ECMs that do not apply to the `electricity` *fuel_type* or switch to `electricity` (via `fuel_switch_to`) are present alongside the `--health_costs` option, only the default version of such ECMs will be prepared and the user will be warned accordingly.

Suppress secondary lighting calculations

`--no_scnd_lgt` suppresses the calculation of the secondary impacts from lighting measures on heating and cooling in commercial buildings, as described in the [Applicable baseline market](#) section.

Captured energy

`--captured_energy` prepares ECM markets and impacts with site-source energy conversion factors calculated using the [captured energy method](#), rather than with the fossil fuel equivalence method as in the default ECM preparation.

Suppress output of ECM-captured energy use

`--no_eff_capt` suppresses the inclusion in the results of the portion of the efficient energy use associated with each ECM that is specifically captured by the ECM (versus remaining with the baseline technology).

Update fields across all ECMs

`--ecm_field_updates` updates fields of all ECMs selected for preparation or included in packages to be prepared. ECM fields are updated using key-value argument inputs. Any number of ECM fields can be updated with additional keys. Keys should correspond to ECM definition fields.

Verbose mode

`--verbose` prints all warning messages triggered during ECM preparation to the console.

1.5.4 Tutorial 4: Modifying the active ECMs list

Prior to running an analysis, the list of ECMs that will be included in that analysis can be revised to suit your interests. For example, if your primary interest is in ECMs that are applicable to commercial buildings, you could choose to include only those ECMs in your analysis.

The “active” (i.e., included in the analysis) and “inactive” (i.e., excluded from the analysis) ECMs are specified in the `run_setup.json` file. There are two ways to modify the lists of ECMs: by *manually editing them* or *using the automatic configuration module*.

If you would like to run your analysis with all of the ECMs and have not previously edited the lists of active and inactive ECMs, you can skip these steps and go straight to [Tutorial 5](#), as all ECMs are included by default.

Tip

As new ECMs are added and pre-processed (by running `ecm_prep.py`), their names are added to the “active” list. Any ECMs that were edited after being moved to the inactive list will be automatically moved back to the active list by `ecm_prep.py`.

Note

When an ECM package is included in the analysis (see [Creating and editing package ECMs](#)), only the package ECM’s name will be added to the “active” list; by default, the names of ECMs that contribute to the package will be added to the “inactive” list to prevent the competition of these contributing ECMs with the package ECM.

Automatic configuration

The automatic configuration module `run_setup.py` can perform a limited set of adjustments to the active and inactive ECM lists in `run_setup.json`.

1. Move ECMs from the active to the inactive list, and vice versa, based on searching the ECM names for matches with user-provided keywords
2. Move ECMs from the active to the inactive list if they do not apply to the climate zone(s), building type, and/or structure type of interest

For each of the changes supported by the module, messages will be printed to the command window that will explain what information should be input by the user. When entering multiple values, all entries should be separated by commas. Any question can be skipped to ignore the filtering option by pressing the “enter” or “return” key.

For the first set of changes, moving ECMs by searching their names for matches with user-provided keywords, the user will be prompted to enter their desired keywords for each move separately, first for moving ECMs from the active to the inactive list, and then for the opposite move. Keyword matching is not case-sensitive and multiple keywords should be separated by commas. Potential inputs from the user might be, for example:

```
ENERGY STAR, prospective, 20%
```

or

```
iecc
```

The user may invert the search for any keyword by adding a “!” character before the search term, for example:

```
iecc, !ENERGY STAR
```

In this case, all ECM names that include “iecc” or do *not* include “ENERGY STAR” will be matched.

To restore all ECMs to the active list from the inactive list, when prompted for the inactive to active move, enter:

```
\s
```

If the user provides keywords for both moves (active to inactive and vice versa) and there are any ECMs that would be picked up by one or more keywords for the moves in each direction, the result would be an ECM being moved from active to inactive and then immediately back to active (or vice versa). For example, if the keyword “prospective” was provided for the move from active to inactive and “heat pump” for the move from inactive to active, an ECM with

the name “Integrated Heat Pump (Prospective)” in either list would be matched by both keywords. To resolve these conflicts, the user would be prompted to decide whether each of these ECMs should end up in the active or inactive lists.

Following these changes, the user will be asked whether additional ECMs should be moved to the inactive list if they are not applicable to the user’s climate zone(s), building type, and/or structure type of interest. For example, a user will be prompted to select the building type (limited to only all residential or all commercial buildings) by number.

```
1 - Residential
2 - Commercial
```

If the user is only interested in residential buildings, they would input

```
1
```

Before running the ECM active and inactive configuration module, it might be helpful to open `run_setup.json` and review the existing list of active and inactive ECMs.

To run the module, open a Terminal window (Mac) or command prompt (Windows) if one is not already open. If you’re working in a new command window, navigate to the Scout project directory (shown with the example location `./Documents/projects/scout-run_scheme`). If your command window is already set to that folder/directory, the first line of the commands are not needed. Run the module by starting Python with the module file name `run_setup.py`.

Windows

```
cd Documents\projects\scout-run_scheme
python run_setup.py
```

Mac

```
cd Documents/projects/scout-run_scheme
python3 run_setup.py
```

If desired, the [manual editing](#) instructions can be used to perform any further fine tuning of the active and inactive ECM lists.

Manual configuration

The `run_setup.json` file specifies whether each ECM will be included in or excluded from an analysis. Like the ECM definition JSON files, this file can be opened in your text editor of choice and modified to change which ECMs are active and inactive.

All of the ECM names should appear in this file under *exactly* one of two keys, “active” or “inactive.” Each of these keys should be followed by a list (enclosed by square brackets) with the desired ECM names. If all ECMs are in the active list, the “inactive” value should be an empty list.

To exclude one or more ECMs from the analysis, copy and paste their names from the “active” to the “inactive” list, and reverse the process to restore ECMs that have been excluded. Each ECM name in the list should be separated from the next by a comma.

Tip

When manually editing the `run_setup.json` file, be especially careful that there are commas separating each of the ECMs in the “active” and “inactive” lists, and that there is no comma after the last ECM in either list.

1.5.5 Tutorial 5: Running an analysis

Once the ECMs have been pre-processed following the steps in [Tutorial 3](#), the uncompleted and competed financial metrics and energy, CO₂, and cost savings can be calculated for each ECM. Competition determines the portion of the applicable baseline market affected by ECMs that have identical or partially overlapping applicable baseline markets. The calculations and ECM competition are performed by run.py following the outline in [Step 3](#) of the analysis approach section.

Note

ECMs prepared via ecm_prep.py with *additional options* may only be simulated in run.py alongside other ECMs that were prepared with the same options and option settings. If discrepancies are found in ECM preparation settings across ECMs in the active list, run.py execution will be halted and the user will see an error message.

To run the uncompleted and competed ECM calculations, open a Terminal window (Mac) or command prompt (Windows) if one is not already open. If you're working in a new command window, navigate to the Scout project directory (shown with the example location `./Documents/projects/scout-run_scheme`). If your command window is already set to that folder/directory, the first line of the commands are not needed. Finally, run run.py as a Python script.

Windows

```
cd Documents\projects\scout-run_scheme
python run.py
```

Mac

```
cd Documents/projects/scout-run_scheme
python3 run.py
```

While executing, run.py will print updates to the command window indicating the current activity – loading data, performing calculations for a particular adoption scenario with or without competition, executing ECM competition, writing results to an output file, and plotting results. This text is principally to assure users that the analysis is proceeding apace. Upon completion, the total runtime will be printed to the command window, followed by an open prompt awaiting another command. The complete competed and uncompleted ECM data are stored in the ecm_results.json file located in the `./results` folder.

Note

On-site electricity generation (from solar PV, fuel cells, and small wind turbines) is now separately reported in ecm_results.json under the **On-site Generation** key. These data encompass on-site electricity generation energy, emissions, and cost projections to 2050 based on the EIA Annual Energy Outlook Reference Case. The reported on-site generation data are limited to the regions and building types covered by the active ECM set in the analysis, and results are reported both overall (under the **Overall** key) and broken out by region and building type (under the **By Category** key).

Tip

The on-site generation results are reported as negative values to facilitate their subtraction from the baseline- and efficient-case measure energy, emissions, and cost results reported in the rest of the ecm_results.json file. To correctly offset these data from the measure results: 1) add the total **On-site Generation** value for a given year to the total baseline-case energy, emissions, or cost value across all measures for the same year to get an adjusted baseline-case value; 2) find the ratio of the adjusted to unadjusted baseline-case values, and; 3) apply the ratio from

#2 to the total efficient-case energy, emissions, or cost value across all measures for the same year to get an adjusted efficient-case value.

Uncompeted and competed ECM results are automatically converted into graphical form by `run.py` (calling `plots.py`). Output plots are organized in folders by *adoption scenario* and *plotted metric of interest* (i.e., `./results/plots/(adoption scenario)/(metric of interest)`). Raw data for each adoption scenario's plots are stored in the XLSX files beginning with "Summary_Data."

Note

On-site generation results are currently not reflected in the graphical results summaries and XLSX write-out.

Additional run options

Users may include additional options alongside the `run.py` command that modify default analysis run settings.

Windows

```
python run.py <additional option> <additional option 2> ... <additional option N>
```

Mac

```
python3 run.py <additional option> <additional option 2> ... <additional option N>
```

The additional run options are described further here.

Results directory

`--results_directory` specifies the directory in which to output results, which includes both raw results and figures. If no directory is provided, results will be outputted to the `./results` folder by default.

Stock/stock cost totals

`--report_stk` reports annual totals for the baseline stock that each ECM could possibly replace, as well as the annual number of those baseline stock units that the ECM has captured after accounting for competition with other ECMs in the analysis. This run option also reports the total and incremental capital cost of all stock units captured by the ECM. Applicable baseline and ECM stock totals are reported by year in `./results/ecm_results.json` under the `Baseline Stock ([units])` and `Measure Stock ([units])` keys, respectively, where `units` differ by technology type. Total and incremental ECM stock costs are reported by year in `./results/ecm_results.json` under the `Total Measure Stock Cost ($)` and `Incremental Measure Stock Cost ($)` keys, respectively.

Market penetration fractions

`--mkt_fracs` reports annual market penetration percentages (relative to the total baseline stock an ECM could possibly replace). ECM market penetration data are reported by year in `./results/ecm_results.json` under the `Stock Penetration (%)` key.

Condensed results data

`--trim_results` limits the results reported in `./results/ecm_results.json` to the avoided energy (Energy Savings (MMBtu)), avoided emissions (Avoided CO₂ Emissions (MMTons)), and avoided energy cost (Energy Cost Savings (USD)) metrics. When this option is selected, the user will also be prompted to optionally select a subset of the full modeling year range to use in reporting results.

Verbose mode

--verbose prints all warning messages triggered during an analysis run to the console.

1.5.6 Tutorial 6: Viewing and understanding outputs

Interpreting results figures

The results figures from the plot generation script `plots.py` are generated for both *adoption scenarios*, and for one of three “metrics of interest”: primary energy use, CO₂ emissions, and energy operating cost. Within the `./results/plots` folder, the folder hierarchy reflects these six cases (two adoption scenarios and three metrics of interest). For each case, the results are presented in three different sets of figures.

1. *Internal rate of return, simple payback, cost of conserved energy, and cost of conserved carbon plotted against a metric of interest.*
2. *A metric of interest aggregated by climate zone, building class, and end use.*
3. *Both uncompleted and completed results for a metric of interest presented separately for each ECM.*

Within each of the plots sub-folders (i.e., `./results/plots/(adoption scenario)/(metric of interest)`), each of these sets of plots is contained within a single PDF file.

Cost-effectiveness figures

The cost-effectiveness figures have file names that begin with “Cost Effective,” followed by the metric of interest and then the adoption scenario (coded as “TP” for technical potential or “MAP” for maximum adoption potential), for example, `Cost Effective Energy Savings-MAP.pdf`.

Each PDF file contains four plots corresponding to the four financial metrics used to assess ECM cost-effectiveness: internal rate of return (IRR), simple payback, cost of conserved energy (CCE), and cost of conserved CO₂ (CCC). For each plot, the applicable financial metric is on the y-axis, and the *reductions* in the metric of interest – energy savings, avoided CO₂ emissions, or energy cost savings – are plotted on the x-axis. All of the data shown include ECM competition and are drawn from a single year, which is indicated in the x-axis label.

Fig. 6 shows an example of the cost-effectiveness figures. All of the cost-effectiveness metrics are plotted against reductions in the metric of interest drawn from a specific year and from one of the two adoption scenarios, 2030 energy savings from the maximum adoption scenario in the case of Fig. 6. The data shown include the effects of *ECM competition*. The various financial metrics can be used to evaluate the potential cost-effectiveness of the ECMs included in an analysis. The relevance of each metric in evaluating ECM cost-effectiveness will depend on the various economic and non-economic (e.g., policy) factors that might impact the particular individual or organization using these results.

On each plot, a dotted horizontal line represents a target threshold for the given financial metric, and the cost-effective region above or below each threshold is highlighted in gray. The thresholds used are positive IRR, five year payback, the projected U.S. average retail electricity price in the year indicated on the x axis, and the *Social Cost of Carbon* in the year indicated on the x axis (2.5% average) for IRR, simple payback, CCE, and CCC respectively. Above or below those lines, depending on whether higher or lower values are more favorable for that metric, the ECM might not be cost-effective in its target market. Total cost-effective savings (from all of the ECMs) for the metric of interest shown in the figure are reported in the top left corner of each plot area. Cost-effectiveness thresholds used in the plots should not be considered “official guidance” or canonical, but rather are provided as a starting point for further investigation of the results. Acceptable thresholds for each financial metric vary by building owner and type, thus each threshold shown is an indication of the range around which an ECM might not be cost-effective.

The shape and fill color of each point indicate the applicable building type(s) and end use(s) for each ECM shown on the plot. Comparing the relative locations of these various points might suggest where there are categories that are more generally cost-effective than others. As with the *baseline market-aggregated figures*, outcomes must be interpreted in light of the particular set of ECMs included in the underlying analysis. In Fig. 6, the top five ECMs pertain to HVAC, water heating, and lighting - each of which constitutes a major end use category with high energy cost savings potential

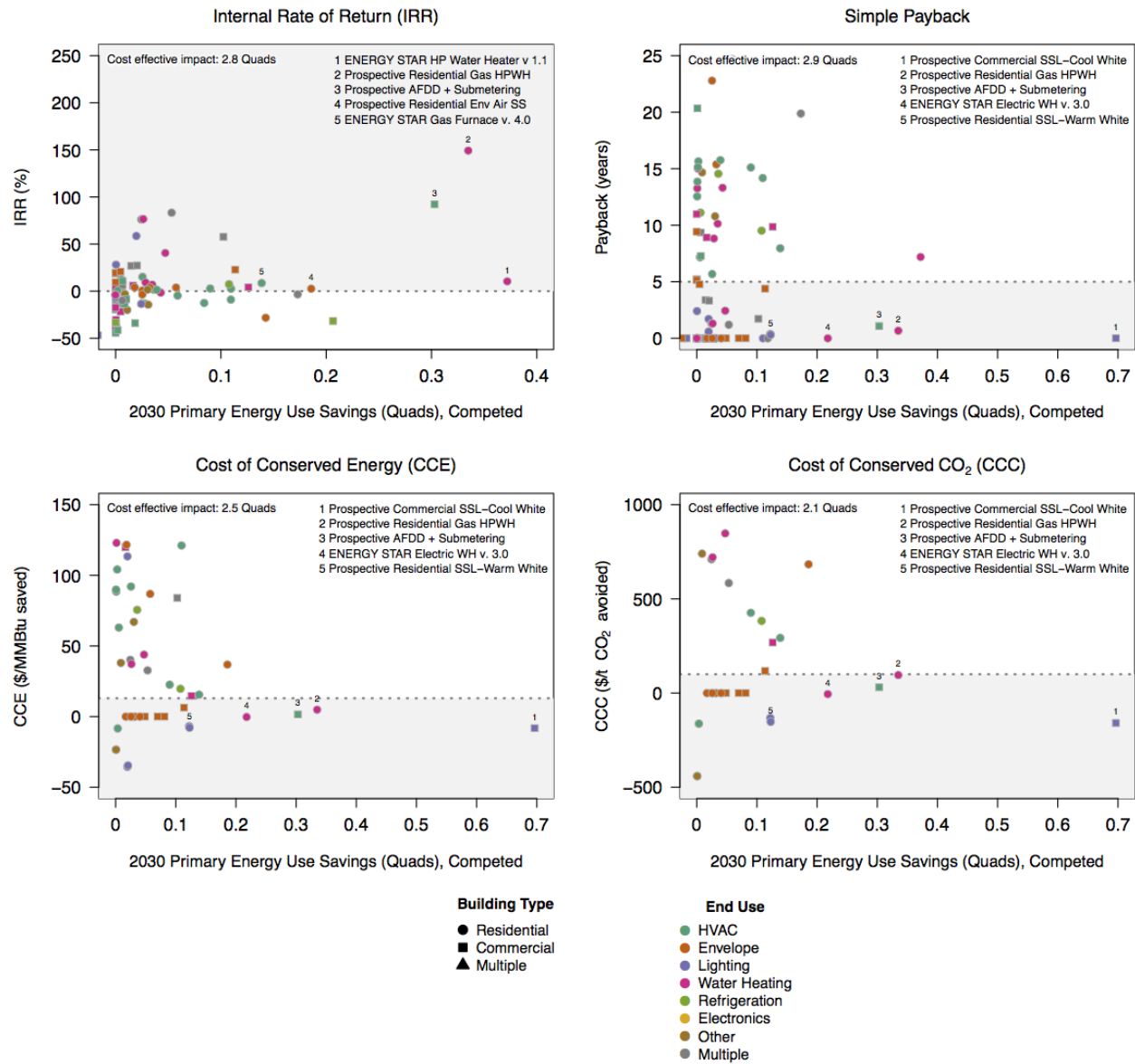


Fig. 6: Each cost-effectiveness figure shows all four cost-effectiveness metrics calculated in Scout for each ECM: internal rate of return, simple payback, cost of conserved energy, and cost of conserved CO₂. For this example figure, data from the maximum adoption potential scenario, inclusive of the effects of ECM competition, are shown. Each point corresponds to a single ECM. ECM building type(s) and end use(s) are indicated by each point according to the legend. For the portfolio of ECMs analyzed for this example, the same ECMs generally appear in the list of cost-effective ECMs with the greatest savings, listed in order of total savings in the top right corner of each plot. Solid-state lighting (SSL) ECMs that appear in the other plots are missing from the internal rate of return plot because their calculated incremental installed cost is zero or near zero. Because those technologies have zero/near zero incremental cost, an internal rate of return either cannot be calculated for them or the calculated value is above the plot region's upper bound. Additionally, several of the ECMs that appear in the internal rate of return and simple payback plots are missing from the cost of conserved energy and CO₂ plots; the latter two metrics have a denominator of competed energy savings and avoided CO₂ emissions, and these values are zero or near zero for the missing ECMs. Total cost-effective savings, based on the thresholds for each cost-effective metric, indicated by dotted horizontal lines, are generally similar.

across the residential and commercial building sectors. Generally speaking, ECMs with a high energy performance to cost ratio are most likely to appear in the top five list.

Baseline market-aggregated figures

The results figures with metrics of interest grouped by the baseline market parameters *climate zone*, building class¹², and *end use* have file names that begin with the metric of interest, followed by the adoption scenario (coded as “TP” for technical potential or “MAP” for maximum adoption potential), and ending with “-Aggregate,” for example, Total Energy Savings-MAP-Aggregate.pdf.

Each PDF contains three plot areas, one for each of the three baseline market parameters. The x-axis corresponds to the modeling time horizon (by default, years 2015 through 2040), and the y-axis corresponds to the *reductions* in the metric of interest – energy *savings*, *avoided* * [CO₂] *emissions*, or energy *cost* * *savings* – indicated in the file name. These plots summarize only the results that account for *ECM competition*. The dotted line on each plot corresponds to the right side y-axis and represents the cumulative results for all the ECMs in the analysis. The line is the same for all three plots within a single PDF. For these figures, while the data are shown as lines instead of points, the data exist as point values for each year in the modeling time horizon and line segments between each year are interpolated and do not represent actual model data.

In general, the figures like Fig. 7 can be used to see at a glance the contributions from various end uses to overall results, as well as the distribution of results among building types and climate zones. While some end uses might appear to contribute more to the total annual or cumulative energy savings (or avoided CO₂ emissions or cost savings), the baseline energy use is different for each end use, and some end uses might appear to contribute more to the savings in part because their baseline energy use is greater. Similarly, while some building types or climate zones might show greater energy savings (or improvement in other metrics) than others, they may also have significantly different baseline energy use.

These results are highly sensitive to the ECMs that are included in the analysis. While an introductory set of ECMs are provided with Scout, if ECMs added by a user are limited to a single end use, for example, it is reasonable to expect that greater reductions in energy use will come from that end use than other end uses.

In the building class plot, differences in the potential savings from new and existing buildings are a result of the building stock being dominated by existing buildings, thus yielding much larger savings in early years of the modeling time horizon than from new buildings.

For most end uses, ECMs and the baseline technologies have similar lifetimes. As the ECMs in that end use diffuse into the equipment and building stock and comparable baseline technology efficiencies improve, the total savings will gradually reach a ceiling or decline year after year, only yielding further increases if more efficient prospective ECMs become available in subsequent years.

In the version of the Fig. 7 that shows energy savings under the technical potential scenario, there are sharp increases in the energy savings in some end uses. These increases come from several key ECMs that have enough of a disparity in energy efficiency from the baseline technologies to yield dramatic overnight savings. Such increases are a result of the ECM adoption assumptions in the technical potential scenario, and should be viewed as an extreme upper bound on the potential for primary energy use reductions in the end uses, climate zones, and building classes shown.

ECM-specific figures

The figures with results for the metric of interest for each ECM included in the analysis all have file names that begin with the metric of interest, followed by the adoption scenario (coded as “TP” for technical potential or “MAP” for maximum adoption potential), and ending with “-byECM,” for example, Total Cost-MAP-byECM.pdf.

The PDF file includes a single plot for each ECM, with the modeling horizon (by default, years 2015 through 2040) on the x-axis and the parameter indicated in the PDF file name on the y-axis – energy, cost, or CO₂ emissions. A legend is included at the end of the figure on the last page of each PDF. Immediately preceding the legend is a summary plot showing the combined effect of all of the ECMs included in the analysis.

¹² Building class corresponds to the four combinations of *building type* and *structure type*.

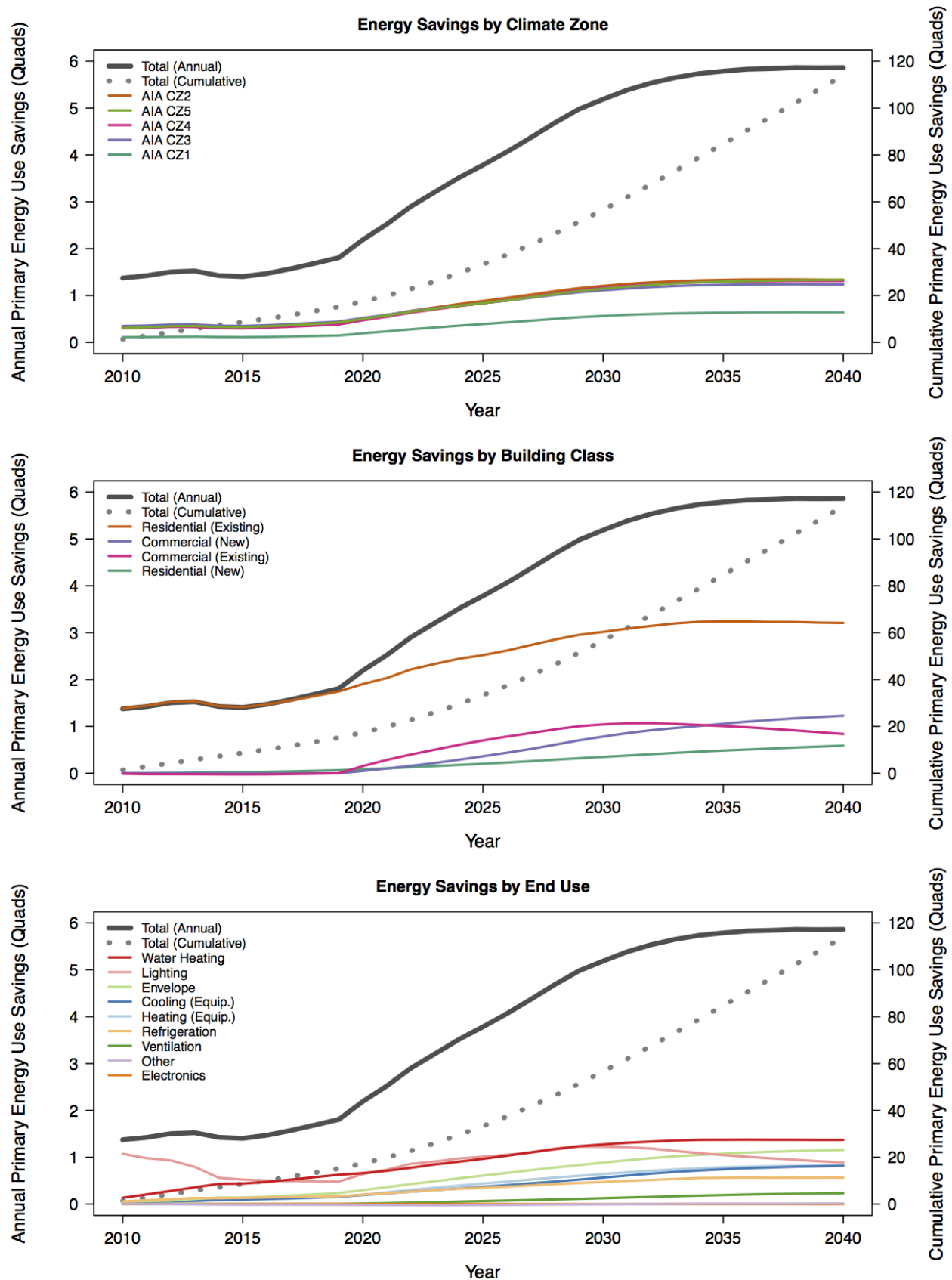


Fig. 7: In this figure, primary energy use reductions are summarized by *end use*, *climate zone*, and building class for the maximum adoption potential scenario, including ECM competition. Data are presented for each year in the modeling time horizon (note: this analysis includes ECMs with 2010 market entry years, plus the start year of 2015 as a default of 2015). The majority of total energy savings in 2010 comes from the introduction of a lighting ECM where the baseline lighting technology has a lifetime of less than one year, as can be seen in the end use breakdown plot. As a result of the short baseline lighting lifetime, the entire lighting stock turns over in 2010. After 2010, annual savings for

The y-axis scale for each plot is adjusted automatically to be appropriate for the data shown. Care should be taken when inspecting two adjacent plots, since what look like similar changes in energy, CO₂, or operating cost values at a glance, might in fact be quite different depending on the y-axes. The y-axis markings must be used to determine the magnitudes in the plots and to compare between plots.

Except for the “All ECMs” plot (illustrated in Fig. 8), each plot shows at least four data series. The two darker, thicker lines correspond to the baseline data with and without ECM competition effects (i.e., “competed” and “uncompeted,” respectively) and represent the portion of all U.S. buildings’ energy use (or CO₂ emissions or operating cost expenses) that could be affected by the introduction of the ECM shown in that plot. The two thinner, lighter lines correspond to the “efficient” results with and without ECM competition effects and reflect the impact of the introduction of the ECM on the baseline energy, CO₂ or operating cost.

These figures are most readily interpreted by comparing relevant pairs of lines.

- **Uncompeted baseline and competed baseline** – the direct or indirect¹³ effects of ECM competition on the total baseline market and associated energy, CO₂, or cost that can be affected by each ECM
- **Uncompeted baseline and uncompeted “efficient”** – the potential for energy savings, cost savings, and avoided CO₂ emissions from the ECM in the absence of alternative technologies that provide the same services
- **Competed baseline and competed “efficient”** – the potential for energy savings, cost savings, and avoided CO₂ emissions from the ECM when other ECMs could provide equivalent service but with different energy/CO₂/cost tradeoffs

In addition to these comparisons, the uncertainty range (if applicable) around “efficient” results and the effect of that uncertainty on competing ECMs should be examined. Fig. 9 illustrates results for an ECM that includes a probability distribution in its definition and the effect of that distribution on related ECMs.

When comparing the uncompeted or competed results in plots like those shown in Fig. 8 and Fig. 9, a difference between the baseline (dark) and efficient (light) cases indicates a potential reduction in the metric of interest plotted. In the absence of competition, the efficient case for both ECMs in Fig. 8 show the immediate realization of the entire savings potential upon market entry (2010 and 2020 for the ENERGY STAR refrigerator and Prospective AFDD ECM, respectively), which is characteristic of the technical potential scenario.

While for many categories, the uncompeted baseline will decline into the future as technology improvements and new standards improve the efficiency of the building and equipment stock, there may be cases where the baseline increases over time. In general, this trend arises due to increases in the size of the stock, increases in home square footage (for residential ECMs), or increases in the capacity or size of the equipment (e.g., increases in the interior volume of refrigerators) that outpace improvements in the performance of the applicable equipment or building envelope component. This type of trend in the baseline appears in Fig. 8 for both the ENERGY STAR refrigerator and Prospective AFDD ECMs. Results for some ECMs show large variations in the baseline for years prior to the current year. These variations are an artifact of the configuration of the National Energy Modeling System (NEMS), which is used to generate the AEO projections.

When ECM competition is introduced, the baseline reflects the year of market entry of each ECM and its competitiveness relative to the other ECMs included in the analysis. Prior to the market availability of an ECM, the competed baseline will reflect the portion of the market left over from competing ECMs. Once the ECM is available, it can begin to recapture some of the market, depending on its competitiveness. If the ECM is more competitive, then it will capture a greater portion of the total available market. If the competed baseline appears to follow the same trend as the uncompeted baseline, the ECM is not capturing any (or any more) of the available market. In Fig. 8, the refrigerator ECM captures all of its potential market in its first year of availability, when no competing refrigerator ECMs are on the market; the potential market for this ECM is substantially reduced in 2015, with the introduction of an updated ENERGY STAR refrigerator ECM, and is all but eliminated in 2020, with the introduction of a prospective refrigerator

¹³ When ECMs are competed against each other, demand-side heating and cooling ECMs that improve the performance of the building envelope reduce the energy required to meet heating and cooling needs (supply-side energy), and that reduction in energy requirements for heating and cooling is reflected in a reduced baseline for supply-side heating and cooling ECMs. At the same time, supply-side heating and cooling ECMs that are more efficient reduce the energy used to provide heating and cooling services, thus reducing the baseline energy for demand-side ECMs. The description of *ECM competition* in Step 3 of the analysis approach section includes further details regarding supply-side and demand-side heating and cooling energy use balancing.

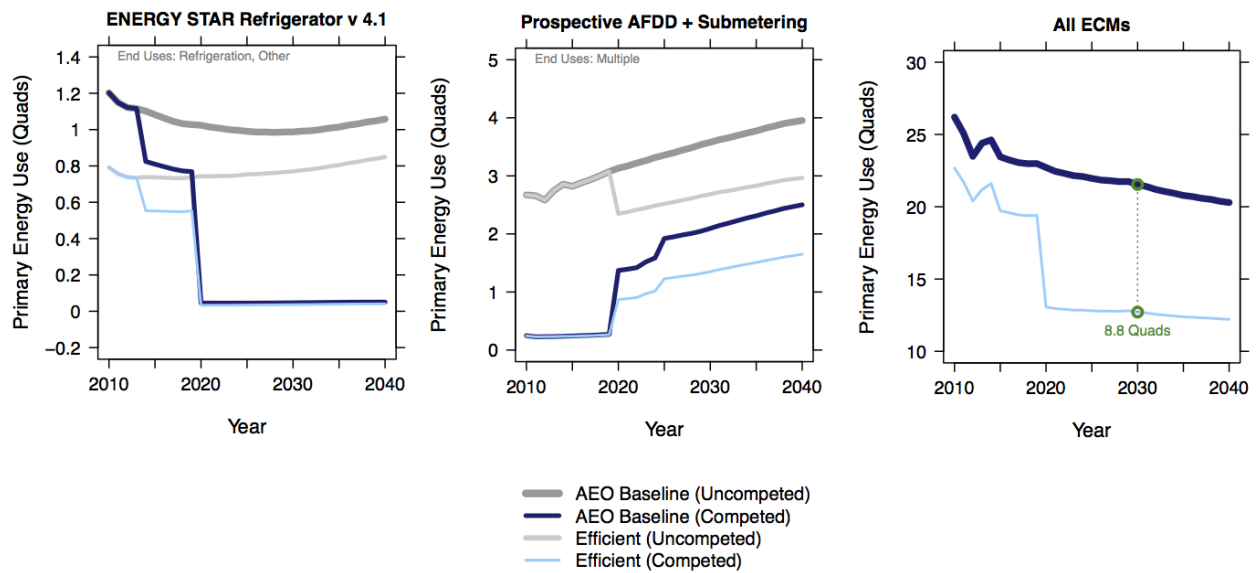


Fig. 8: Primary energy use baselines, and improvements with the adoption of two ECMs – ENERGY STAR Refrigerator v. 4.1 and Prospective AFDD + Submetering – are shown for each year in the modeling time horizon (note: the ENERGY STAR refrigerator ECM has a 2010 market entry year; thus the start year is reset to 2010 from a default of 2015). The data shown are from the *technical potential* adoption scenario, which is reflected in the abrupt single-year changes in energy use when the ECM enters the market. The data are derived from a model that included many ECMs besides those shown, thus the ECMs’ impacts change under *competition*. Affected end uses are shown at the top of each plot to help determine which ECMs might be competing with each other. The “All ECMs” figure on the right side shows the aggregate reductions in the metric of interest from all of the ECMs included in the analysis, and the dotted line at the year 2030 indicates the total savings in that year.

ECM that satisfies aggressive cost and performance targets (competing refrigerator ECMs not shown). Conversely, the potential market for the Prospective AFDD ECM in Fig. 8 grows after its market introduction; this growth continues between 2020-2025, when the introduction of additional ECMs reduces the market share of ECMs that compete with the Prospective AFDD ECM, thus increasing its potential market.

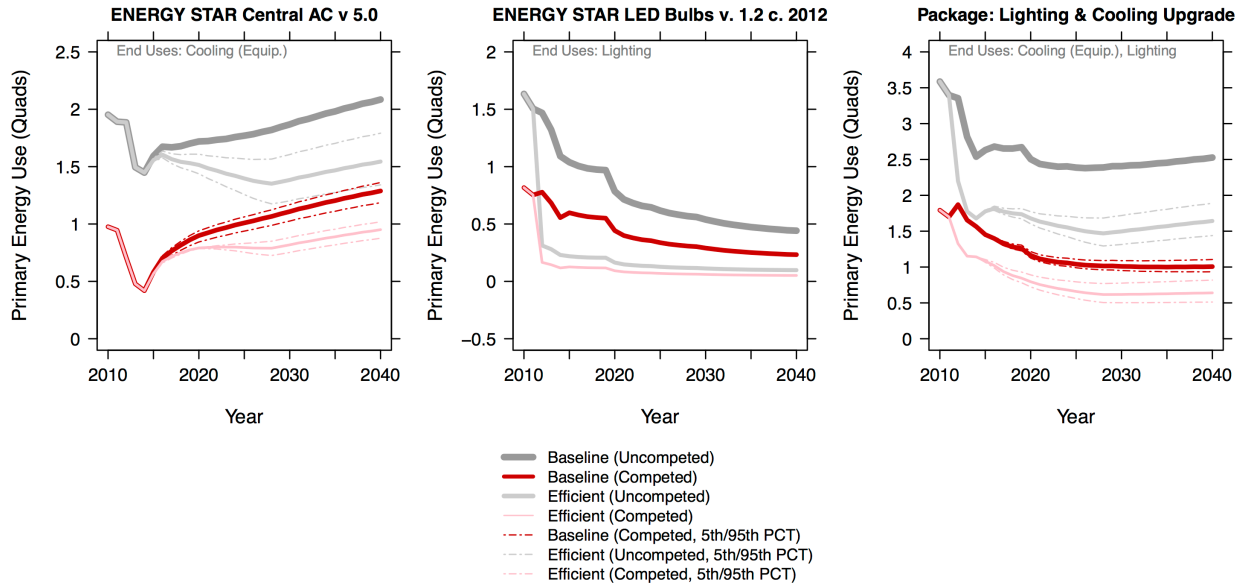


Fig. 9: The ENERGY STAR Central AC ECM includes a probability distribution on one or more of the installed cost, efficiency, or product lifetime, which is reflected in the results. The 5th and 95th percentile bounds for each affected value are shown with dashed lines of the same color. The two ECMs shown on the left are combined into the package ECM on the right. The package ECM results reflect the effect of the probability distributions in the contributing ECMs.

The effect of probability distributions are reflected in the results of the ECMs that include them in their definitions, as well as in the competed results for any competing ECMs and in any packages that include the ECM. The effects of uncertainty will always appear in the results with the ECM applied (i.e., the “efficient” results), though if the spread of the distribution is quite small, it might be difficult to see the dashed lines showing those effects in plots Fig. 9. In addition, if a probability distribution is applied to the installed cost or energy efficiency of the ECM, it will create uncertainty in the capital and/or operating cost of the ECM, and thus will introduce uncertainty in *ECM competition*, which is reflected in the competed baseline for the ECM. The ENERGY STAR air conditioning ECM in Fig. 9 shows this effect.

Viewing tabular outputs

The plot generation script `plots.py` also produces Excel-formatted files containing summaries of the results. The summary results for each adoption scenario are stored in the corresponding scenario folder within the `./results/plots` directory. The structure of the results in the files corresponding to each scenario is identical. Each file has three tabs, corresponding to energy use, CO₂ emissions, and energy cost results. These results correspond to the data that are shown in the ECM-specific plots, as in Fig. 8, and the tabular results can be used to create custom visualizations different from those automatically generated with `plots.py`.

 Tip

If you are experienced with matplotlib, you can also modify plots.py to tailor the figures to your preferences.

On each tab, the first five columns provide information about the ECM and the type of data reported in each row. The first column contains the name of the ECM for the data in each row and the third through fifth columns provide details regarding the climate zones, building classes, and end uses that apply to each ECM. The second column indicates the type of data in each row – one of the four series shown in Fig. 8, the baseline and efficient results, with and without ECM competition. The sixth through ninth columns contain results for the financial metrics: internal rate of return (IRR), simple payback, cost of conserved energy (CCE), and cost of conserved CO₂ (CCC). When any of the financial metrics cannot be calculated (e.g., simple payback for a negative incremental capital cost, or negative energy cost savings) the metric will be reported as the value 999. The columns beyond the ninth column contain the results for the metric of interest (energy use, CO₂ emissions, or energy cost) indicated by the worksheet tab name. Each of those columns corresponds to a year in the simulation, with the year indicated in the first row.

For a given set of results data on a single tab, each ECM included in the simulation appears in four rows that are distinguished by the ECM's name (listed under the “ECM Name” column). These four rows correspond to the uncompleted and competed baseline results, as well as the (“efficient”) results with the ECM applied, again with and without competition. For each ECM, these rows correspond to the four primary lines that appear in the ECM-specific results figures, as in Fig. 8.

In each results tab, rows 2-22 include results summed across the entire ECM portfolio (rows with “All ECMs” listed under the “ECM Name” column). The information in these 21 rows is interpreted as follows: rows 2-3 show competed baseline and efficient results summed across the entire ECM portfolio - these rows correspond to the two primary lines that appear in the “All ECMs” plot in the ECM-specific results figures; row 4 shows the difference between the competed baseline and efficient results shown in rows 2-3 (“Baseline - efficient”); and rows 5-22 break down the result shown in row 4 by climate zone (5 climate categories/rows), building type (4 building type categories/rows), and end use (9 end use categories/rows).

 Note

For each ECM in the results, in addition to the *total* energy use, CO₂ emissions, and energy cost results contained in the Excel files, the *ecm_results.json* file includes those results broken out by each of the applicable baseline market parameters – *climate_zone*, *bldg_type*, *structure_type*, *fuel_type*, *end_use*, and *technology* – that apply to each ECM. These results breakdowns are provided for both the baseline and efficient cases (without and with the ECM applied, respectively).

1.6 Analysis Approach

A full analysis in Scout has three primary steps, which are detailed further in the sections below.

1. Develop an initial ECM definition.
2. Finalize ECM definition by retrieving additional performance and market data.
3. Simulate ECM impact across multiple adoption scenarios, with and without ECM competition.

1.6.1 Step 1: Develop initial ECM definition(s)

Users may define entirely new ECMs or modify existing ECM definitions located in the *./ecm_definitions* directory. Each Scout ECM definition centers around seven user-specified attributes. Three of these seven attributes – energy efficiency, installed cost, and lifetime – may be assigned a probability distribution¹ instead of a point value; these

¹ Currently supported distributions: *uniform*, *normal*, *lognormal*, *triangular*, *weibull*, and *gamma*.

attributes also require source information from the user.

Acceptable sources for ECM input definitions include: [RSMeans](#) (for installed cost); databases from [EIA](#), [ENERGY STAR](#), and national labs; peer-reviewed journal publications or reports; and product literature.

Selections for the applicable baseline market parameters are used in [step 2](#) to retrieve total energy use, CO₂, and operating cost data for the baseline market. The ECM can then be applied to that baseline market to calculate savings impacts. The specific values available for all of the applicable baseline market parameters are included in the [Applicable baseline market options](#) section.

- Applicable baseline market parameters
 - Climate zone (correspond to the five [AIA climate zones](#)).
 - Building type (correspond to [three residential](#) and [eleven commercial](#) building types used in the EIA Annual Energy Outlook (AEO)).
 - Building vintage (“new” and/or “existing”).
 - Fuel type (“electricity”, “natural gas”, “distillate”, or “other”).
 - End use (correspond to the major [residential](#) and [commercial](#) energy end uses defined in the AEO).
 - Technology name (for envelope components, technology names correspond to the names of [residential](#) and [commercial](#) thermal load components; for building equipment, technology names generally correspond to major equipment types used in the [AEO](#)²).
- Market entry and exit year
 - Market entry year reflects the first year the ECM is commercially available, while market exit year typically reflects a future efficiency standard that would render the ECM obsolete.
- Energy efficiency
 - Energy efficiency is specified at the unit level in absolute terms (e.g., U-value and solar heat gain coefficient for a window, or COP for a heat pump) using the [required units](#) or as a percentage relative savings value.
- Installed cost
 - Product or technology [installed cost](#) should be specified in absolute (not incremental) terms.
 - Cost units vary by building sector and technology type, as specified in the [ECM Definition Reference](#).
- Lifetime
 - Expected lifetime of the ECM in years.
- Additional ECM features
 - Is the ECM a service add-on or replacement?
 - * ECMs may either directly replace the service of a comparable “business-as-usual” technology (e.g., a more efficient air source heat pump) or enhance the efficiency of this incumbent technology (e.g., a window film or HVAC controls retrofit).
 - * The choice of whether an ECM is of the “replacement” or “add-on” type has implications for cost calculations. In the former case, the ECM’s incremental installed cost is calculated relative to that of the comparable baseline unit; in the latter case, the baseline cost is zero and the ECM’s incremental installed cost is equal to its installed cost.
 - Does the ECM require fuel switching?

² Note that this document does not cover lighting, where varying bulb types are used, or Miscellaneous Electric Loads (MELs), which are not broken into specific technologies in the Annual Energy Outlook.

- * In cases where an ECM has a different fuel type than a comparable “business-as-usual” technology (e.g., an electric cold climate heat pump replacing a gas furnace for heating service), the user must specify the fuel type switched to (e.g., “electricity”).
- Time sensitive valuation of energy efficiency
 - * Optionally, a user may specify one or more ECM features that apply efficiency impacts differently according to time of day and season, using hourly baseline energy load shapes, electricity costs, and average CO₂ emissions factors.

1.6.2 Step 2: Finalize ECM definition

ECM definitions from [step 1](#) are finalized using the `ecm_prep.py` module. The total (stock-wide) energy use, CO₂ emissions, and operating costs of the ECM are calculated for baseline and efficient cases, not accounting for ECM competition.

Calculating total baseline energy, CO₂, and cost (uncompeted)

- Total uncompeted energy use, CO₂ emissions, and operating cost baselines are calculated for each ECM from a 2013-2050 projection of U.S. building stock, energy use, and unit characteristics. These baseline data are mostly drawn from the inputs and outputs of the EIA [Annual Energy Outlook \(AEO\) reference case](#) simulations.³ Where AEO data are not available, such as for building envelope component and electronics technologies, BTO develops original datasets using multiple sources external to DOE.
 - Baseline stock data represent the total number of units of a certain incumbent or “business-as-usual” technology associated with a given baseline market and year in the projection period. An example is the number of air-source heat pump units in all existing single family homes in mixed dry climates in the year 2020. When a number of units value is not available or not applicable for a baseline market, such as for building envelope component technologies, total building floor area square footage associated with that baseline market and year is used to quantify the baseline stock.
 - Baseline energy use data represent the total energy use attributed to a certain baseline market and year in the projection period. For example, the energy used to provide heating in all existing single family homes in mixed dry climates in the year 2031.
 - Baseline technology characteristics data represent the primary attributes of an incumbent or “business-as-usual” building technology, namely the technology’s energy efficiency (in absolute units, e.g., COP), installed cost, and lifetime. Additionally, these data include consumer choice parameters for each technology, which are used for ECM competition (see [step 3](#)).
- Once baseline energy use numbers are established for each technology, these energy use numbers must be translated from site to source (or “primary”) energy using site-source conversion factors calculated from the electricity and electricity related losses data in the [AEO energy consumption by sector and source table](#). [Fuel-specific energy costs](#) and CO₂ emission intensities, calculated by dividing fuel-specific CO₂ emissions by fuel-specific energy use, are also derived from AEO summary tables. CO₂ emissions costs are drawn from the most recent U.S. Office of Management and Budget [Social Cost of Carbon](#) estimates⁴.

Calculating total efficient energy, CO₂, and cost (uncompeted)

- Total uncompeted energy use, CO₂ emissions, and operating cost baselines calculated for an ECM from the input data are used to generate the total uncompeted energy use, CO₂ emissions, and operating costs with the ECM implemented - herein referred to as the “efficient” case - as follows:
 - calculate an efficient energy fraction for the ECM; this is the fraction of per unit energy use under a full ECM implementation compared to the per unit energy use of a baseline case with no ECM implementation,

³ EIA provides detailed documentation on the assumptions of the National Energy Modeling System (NEMS) it uses to project [residential and commercial](#) sector energy use out to 2050 for the AEO.

⁴ Data derived from Table A1, assuming a 3% average discount rate.

- multiply the efficient energy fraction by the ECM’s total baseline energy use to yield an efficient energy use total, and
- use the ECM’s total efficient energy use to calculate its total efficient CO₂ emissions and operating costs, using CO₂ emission intensities, fuel-specific energy cost data, and the [Social Cost of Carbon](#).
- The magnitude of the difference between an ECM’s total baseline and efficient energy, CO₂, and operating costs depends on what portion of the ECM’s baseline market it can affect in each year of the projected time period. This available market portion is determined by [stocks-and-flows](#) in the baseline markets over time. The primary stock-and-flow variables accounted for in Scout are listed here.
 - Stock variables (measured at a point in time, i.e., in a given year)
 - * Competed stock – defined as the number of technology units that are new or up for retrofit/replacement in a given year.
 - * Non-competed stock – defined as the number of existing technology units not up for retrofit/replacement.
 - Flow variables (measured over a period of time, i.e., from year to year)
 - * New additions – defined as the technology units associated with new building construction; it is a fraction of the total existing technology units, determined by the ratio of new buildings to total buildings (residential) or new square footage to total square footage (commercial). The NEMS [residential](#) and [commercial](#) sub-module documentation outlines the derivation of the new and total buildings stock and square footage data.
 - * Retrofits – defined as the technology units up for replacement before the end of their useful lifetime; it is a fraction of the total existing technology units, determined based on available literature on typical [residential](#) and [commercial](#) building equipment retrofit rates.
 - * Replacements – defined as the technology units at the end of their useful lifetime; it is a fraction of the total existing technology units, determined by 1/lifetime of the existing technology, excepting technical potential cases (see next bullet).
- ECM diffusion into baseline markets is modeled under technical potential and maximum adoption potential scenarios. In both cases, the available market portion is entirely captured by ECMs (e.g., no competed stock remains with a “business-as-usual” technology).⁵ The primary difference between the two scenarios is in how they define competed stock for the first year of ECM market entry:
 - in the technical potential case, an ECM competes for all new and existing stock in its market entry year, while
 - in the maximum adoption potential scenario, an ECM competes for all new stock and existing stock that is up for retrofit or replacement in its market entry year.
- Under these two scenarios, the portions of a baseline market that are captured by an ECM never return to a “business-as-usual” technology option. In a technical potential case, this means the complete market saturation an ECM secures upon market entry is sustained across the entire projection period.
- Because baseline markets are comprised mostly of existing stock (typically 97% or more in each year) and existing stock retrofit rates are low (less than 2% in each year), the market diffusion of an ECM in a maximum adoption potential scenario is mostly driven by existing stock replacement rates. As shown in the maximum adoption scenario results of [Fig. 10](#), replacement of technologies with short lifetimes yields immediate market saturation for an ECM ([Fig. 10a](#)) and no difference between the technical potential and maximum adoption potential scenario results, while replacement of technologies with moderate to long lifetimes yields more gradual diffusion of an ECM into its baseline market ([Fig. 10b](#) and [Fig. 10c](#)).

⁵ This assumption reflects our current inability to reliably determine how a consumer might choose between an ECM and updated version of a “business-as-usual” technology. A future version of Scout may apportion some of an available market portion to this updated “business-as-usual” technology, under an “Adjusted Adoption Potential” scenario.

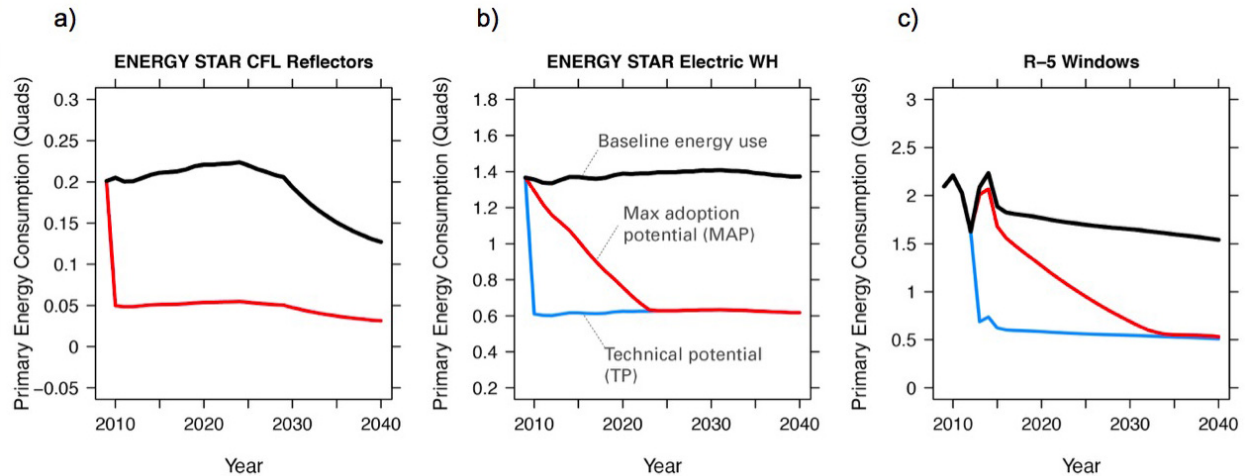


Fig. 10: Scout ECMs are applied to a baseline energy market under two technology diffusion scenarios: maximum adoption and technical potential. In a maximum adoption case, diffusion rates depend on realistic rates of new construction and retrofits in a given year, as well as the rate of turnover in the existing baseline equipment that an ECM could replace. The three ECMs shown represent (from left to right) fast, moderate, and slow diffusion rates under a maximum adoption scenario.

- While the total baseline and efficient ECM energy, CO₂, and operating costs calculated in this step account for stocks-and-flows, they do not account for competition across multiple ECMs for the same baseline market. ECM competition is handled in *step 3*.

1.6.3 Step 3: Simulate ECM impact

The final step, contained in the `run.py` module, calculates each ECM's total energy savings, avoided CO₂ emissions, and operating cost savings impacts based on the total uncompleted energy use, CO₂ emissions, and operating costs calculated in *step 2*. Cost savings impacts are used to calculate per-unit financial metrics for the ECMs. Here, both competed and uncompleted ECM impacts and financial metrics are calculated.

Calculating uncompleted ECM energy savings and financial metrics

- Uncompleted ECM energy savings, avoided CO₂ emissions, and operating cost savings impacts are calculated by subtracting the total uncompleted efficient energy, CO₂, and operating costs calculated in *step 2* from total uncompleted baseline energy, CO₂, and operating costs calculated in *step 2*. Note that each of these total figures are calculated for technical potential and maximum adoption potential scenarios, and therefore ECM impacts and financial metrics are also described in terms of these scenarios.
- ECM financial metrics are calculated by normalizing ECM savings impacts to the total number of stock units and comparing unit savings to the ECM's incremental capital cost over the comparable "business-as-usual" technology.
 - Internal Rate of Return (IRR) is the discount rate that balances the net present value of the ECM cost (negative cash flow) against the savings realized by the ECM on a per-unit basis (positive effective cash flow).
 - Simple Payback Period divides the per-unit cost of the ECM by its per-unit annual energy savings compared to the "business-as-usual" unit.
 - Cost of Conserved Energy (CCE) divides the per-unit cost of the ECM by its discounted⁶ per-unit lifetime savings compared to the "business-as-usual" unit. In one variant of the CCE calculation, discounted lifetime

⁶ We use a default real discount rate of 7%, based on the Office of Management and Budget's *Guidelines and Discount Rates for Benefit-Cost Analysis of Federal Programs*, p. 9 ("Base-Case Analysis").

cost savings from avoided CO₂ emissions are added to the numerator of the calculation, using [Social Cost of Carbon](#) estimates as a carbon tax.

- Cost of Conserved CO₂ (CCC) follows the same calculation as CCE, but uses avoided CO₂ emissions in the denominator and energy cost savings in the numerator (if applicable).

Competing ECMs and updating savings

- ECMs with overlaps in their applicable baseline markets compete for the overlapping portions of these markets on the basis of their cost effectiveness from a consumer perspective. In general, ECMs with lower incremental capital costs and higher operational cost savings across their lifetimes capture larger portions of the overlapping baseline markets.
- For example, R-5, R-7, or R-10 window ECMs could each replace the same “business-as-usual” window technology. The initial savings impacts calculated for each of these ECMs will be based on the entire applicable baseline market. Those savings impacts must be scaled by the share of the baseline window market each ECM is modeled as capturing to avoid double counting of savings. Assuming the R-7 window is most cost effective and R-10 is least cost effective, the market shares might be R-5, 35%; R-7, 45%; and R-10, 20%.
- This use of market shares to reflect ECM competition ensures that competing ECMs with similar levels of cost effectiveness will have similar savings impacts after adjusting for competition.
- In general, ECM competition calculations in Scout weigh an ECM’s annualized capital and operating costs against the capital and operating costs for competing ECMs to determine each ECM’s competed market share. However, the specific calculation steps differ somewhat between the residential and commercial building sectors.
- Once ECM market shares are determined, uncompleted ECM savings impact estimates are multiplied by these market shares to arrive at competed ECM energy savings, avoided CO₂ emissions, and operating cost savings impacts.
- For heating and cooling ECMs, a post-competition calculation is needed to ensure that total supply-side heating/cooling energy use (e.g., as consumed by a heat pump, or furnace) equals total demand-side heating/cooling energy use (e.g., that attributable to heat transfer through the envelope and other thermal load components).

ECM-specific results from the analysis of the portfolio of ECMs

- Filter variables summarize an ECM’s applicable climate zone(s), building class(es), and end use(s).
- Baseline and efficient results summarize an ECM’s total baseline and efficient energy use, CO₂, emissions and operating costs, as well as the savings realized by comparing the efficient case to the baseline case. Baseline and efficient results are reported as totals for the ECM and also broken down by building sector (residential/commercial), climate zone, and end use⁷.
- Financial metrics summarize an ECM’s *financial metrics*.
- If applicable, average and 5th/95th percentile values are reported for all efficient markets, savings, and financial metrics outputs to accommodate ECM input uncertainty analysis.

1.7 Heating and Cooling Load Components

This section describes the purpose and derivation of the thermal load components files for [residential](#) and [commercial](#) buildings.

⁷ End use categories for the markets and savings are not the same as the AEO end uses. These end use categories are similar to the 2015 DOE Quadrennial Technology Review, Figure 5.1, but with the “Drying” end use lumped together with “Other.”

1.7.1 Overview

Energy conservation measures (ECMs) evaluated by Scout yield savings on either the supply or the demand side of energy consumption. As an example of the former, consider the replacement of electric heating and cooling equipment with a more efficient heat pump: in this case, the new heat pump reduces the amount of electricity needed to provide a fixed amount of heating/cooling to a building space. As the absolute heating/cooling *load* is unchanged, the ECM's energy savings come from the more efficient manner in which the heating/cooling is supplied.

Conversely, a demand side ECM reduces the absolute load that must be met for a given end use, leaving the supply for that end use unchanged. An example of such a ECM is highly insulating window replacements: in offering improved resistance against thermal conduction from the space to the outdoors, the replacement reduces the amount of heating energy that must be added to the space to maintain a comfortable air temperature, allowing demand side energy savings.

This section focuses on the evaluation of demand side ECMs for the heating and cooling end uses. Here, we might be interested how much national heating and/or cooling energy could be saved through the highly insulating window replacements previously mentioned, or by other ECMs relating to the building envelope such as adding insulation or shading devices, amongst many others. In these cases, estimations of energy savings require an understanding of how a building's heating/cooling energy use is partitioned. For example, were we to evaluate the heating energy savings potential of highly insulating windows, we would first need to know the amount of heating energy that is lost through a building's windows to the outdoors (which the insulating window would reduce). This figure is obtained by multiplying a building's total heating energy use by the percentage of heating load attributable to conduction losses through windows, referred to subsequently as the heating load "component" for window conduction.

While the energy use part of this calculation is easily drawn from the U.S. Energy Information Administration (EIA) [Annual Energy Outlook \(AEO\)](#), establishing heating and cooling load component values for a range of building types and geographical regions is less straightforward. To improve the transparency of this process, the section below outlines the steps that were used to derive a new set of thermal load components for each of the AEO residential and commercial building types, across all nine U.S. census regions. This approach yields two files, [Res_TLoads_Final.txt](#) and [Com_TLoads_Final.txt](#).

Thermal load components data are drawn in large part from two simulation studies of residential and commercial buildings by [Huang, Hanford, and Yang \(1999\)](#) and [Huang and Franconi \(1999\)](#), respectively. The residential study uses parametric computer simulations of 112 single-family and 63 multi-family residential building prototypes to quantify the contributions of building components such as roofs, walls, windows, infiltration, outside air, lighting, equipment, and people to the aggregate heating and cooling loads in U.S. residential buildings; the commercial study does the same for aggregate commercial heating and cooling loads using computer simulations of 120 commercial building prototypes.

1.7.2 Deriving heating and cooling load components

The residential and commercial building thermal load component values are derived from the data in the simulation study publications.

1. Identify relevant tables in the residential and commercial simulation studies.
 - Residential: ResStock: Annual Baseline Results with Component Loads CSV.
 - Simulations cover all three AEO residential building types and nine U.S. census regions, for heating and cooling end uses separately.
 - Baseline tables include unnecessary sub-groupings of residential thermal loads data, including by building vintage and for multiple cities within each census division.
 - Each simulated building type/location combination has an associated number of buildings in the U.S. stock reported with it for weighting purposes.
 - **Map residential table building types to AEO building types.**
 - * Single-Family Detached -> single family home
 - * Single-Family Attached -> single family home

- * Mobile Home -> mobile home
 - * 50 or more Unit -> multi family home
 - * 20 to 49 Unit -> multi family home
 - * 10 to 19 Unit -> multi family home
 - * 5 to 9 Unit -> multi family home
 - * 3 or 4 Unit -> multi family home
 - * 2 Unit -> multi family home
 - * nan -> None
- Commercial: Annual Baseline Results with Component Loads CSV.
 - Simulations cover nine of eleven AEO commercial building types and nine U.S. census regions for heating and cooling end uses separately.
 - Each simulated building type/location combination has an associated floor space square footage in the U.S. stock reported with it for weighting purposes.
 - **Map commercial table building types to AEO building types.**
 - * Hospital -> Healthcare
 - * Outpatient -> Healthcare
 - * Retail Strip mall -> Merch./Service
 - * Retail Standalone -> Merch./Service
 - * Full Service Restaurant -> Merch./Service
 - * Medium Office -> LargeOffice
 - * Quick Service Restaurant -> Food Service
 - * Large Hotel -> Lodging
 - * Secondary school -> Education
 - * PrimarySchool -> Education
 - * Grocery -> Food Sales
 - * LargeOffice, SmallOffice, Warehouse -> Same in AEO
2. Condense CSV tables into final set of thermal load components data.

Python pseudocode (Residential)

1. Import results_up00_update CSV file.
2. Find subset of CSV rows associated with each unique combination of census division (residential), AEO building type, and heating or cooling end use.
3. For each subset of rows, calculate a weighted average of the thermal load components across all rows using the number of buildings (residential) associated with each row to establish weighting factors.
4. Combine the thermal load components calculated for each unique combination of census, AEO building type, and end use into a master table.
5. Write the final thermal load components table to a text file.

Python pseudocode (Commercial)

1. Import upgrade0_agg CSV file.

2. Find subset of CSV rows associated with each unique combination of census division, AEO building type, and heating or cooling end use.
3. For each subset of rows, calculate a weighted average of the thermal load components across all rows using the number of buildings associated with each row to establish weighting factors.
4. Combine the thermal load components calculated for each unique combination of census, AEO building type, and end use into a master table.
5. Calculate thermal load components for missing AEO building types using a weighted combination of similar and available ComStock building types.
 - The missing “Assembly” commercial building type is created as a floor area weighted combination of “Secondary School”, “Small Office”, and “Merch/Service” building types.
 - The missing “Other” commercial building type is created as a floor area weighted combination of all building types.
6. Write the final thermal load components table to a text file.
3. [Res_TLoads_Final.txt](#) and [Com_TLoads_Final.txt](#) files contain final thermal loads components broken down by census division, AEO building type, and heating/cooling end use for further analysis.

1.7.3 References

Speake A, Wilson EJ, Zhou Y, Horowitz S. (2023). Component-level analysis of heating and cooling loads in the US residential building stock. *Energy and Buildings*, 299, 113559.

Huang J, Franconi E. (1999). Commercial heating and cooling loads component analysis (LBNL-37208). Berkeley, CA: Lawrence Berkeley National Laboratory.

1.8 ECM Definition Reference

In some parts of the definition of an ECM, specific values must be entered for the ECM to be valid and successfully included in an analysis. In particular, the installed cost and energy efficiency units used must match exactly with the expected units for a given building sector, end use, and technology type. These units are defined by the EIA Annual Energy Outlook data used to define the baseline technology cost, efficiency, and lifetime.

1.8.1 Applicable baseline market options

For each of the keys in the applicable baseline market definition, only specific entries are expected and allowable. This section outlines those acceptable entries for each of the keys.

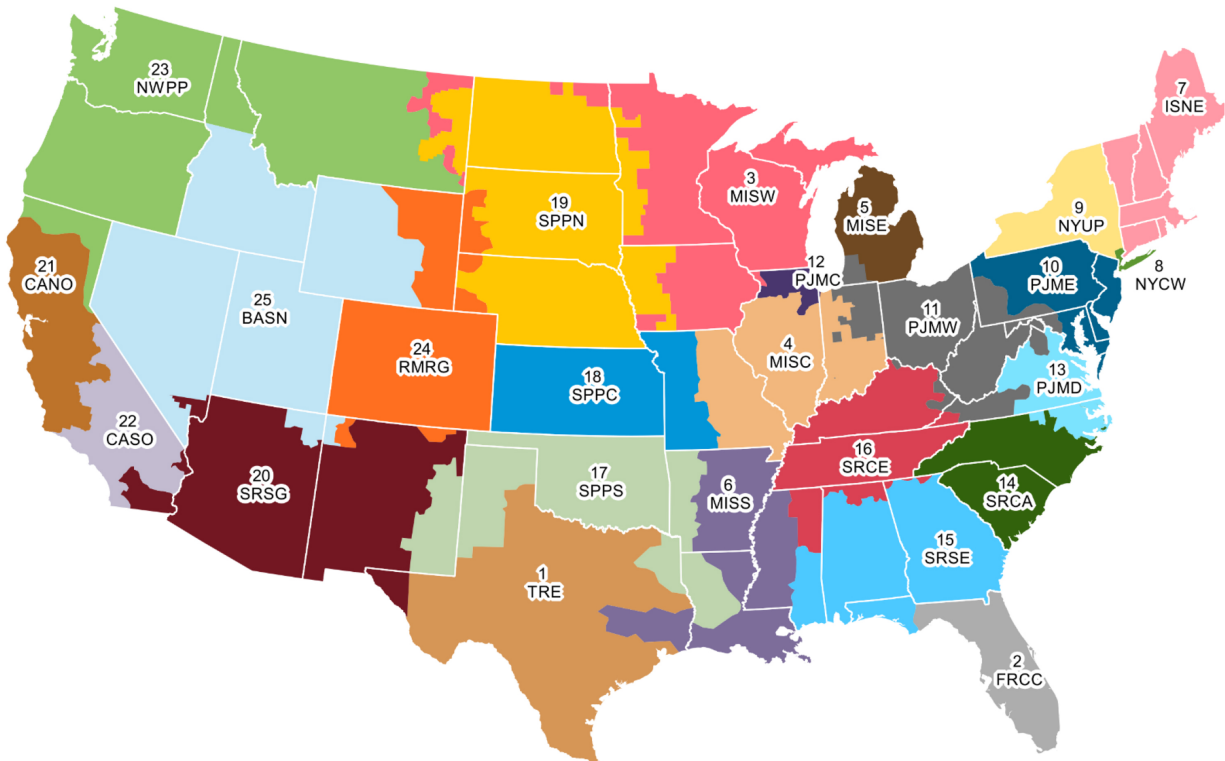
For some keys, there are shorthand summary values that can be used when all or a large group of more specific values for that key apply. For example, if all of the climate zones should be included in the applicable baseline market, the value “all” can be specified instead of typing out each climate zone name in a list. These shorthand values are provided after the semi-colon in the lists below. Additional notes might also be provided to further clarify the different summary values available for a given key. More information regarding the use of these shorthand values is in the [Baseline market shorthand values](#) section.

Climate zone

EIA Electricity Market Module (EMM) regions

TRE*, **FRCC**, **MISW**, **MISC**, **MISE**, **MISS**, **ISNE**, **NYCW**, **NYUP**, **PJME**, **PJMW**, **PJMC**, **PJMD**, **SRCA**, **SRSE**, **SRCE**, **SPPS**, **SPPC**, **SPPN**, **SRSR**, **CANO**, **CASO**, **NWPP**, **RMRG**, **BASN**; all, warm climates, cold climates

The bolded EMM regions are categorized as “warm climates;” the remainder are “cold climates.”



Region ID	NERC/ISO subregion	Geographic Name*
1- TRE	Texas Reliability Entity	Texas
2- FRCC	Florida Reliability Coordinating Council	Florida
3- MISW	Midcontinent ISO/West	Upper Mississippi Valley
4- MISC	Midcontinent ISO/Central	Middle Mississippi Valley
5- MISE	Midcontinent ISO/East	Michigan
6- MISS	Midcontinent ISO/South	Mississippi Delta
7- ISNE	NPCC/ New England	New England
8- NYCW	NPCC/NYC & Long Island	Metropolitan New York
9- NYUP	NPCC/Upstate NY	Upstate New York
10- PJME	PJM/East	Mid-Atlantic
11- PJMW	PJM/West	Ohio Valley
12- PJMC	PJM/Commonwealth Edison	Metropolitan Chicago
13- PJMD	PJM/Dominion	Virginia

Region ID	NERC/ISO subregion	Geographic Name*
14- SRCA	SERC Reliability Corporation/East	Carolinas
15- SRSE	SERC Reliability Corporation/Southeast	Southeast
16- SRCE	SERC Reliability Corporation/Central	Tennessee Valley
17- SPPS	Southwest Power Pool/South	Southern Great Plains
18- SPPC	Southwest Power Pool/Central	Central Great Plains
19- SPPN	Southwest Power Pool/North	Northern Great Plains
20- SRSG	WECC/Southwest	Southwest
21- CANO	WECC/CA North	Northern California
22- CASO	WECC/CA South	Southern California
23- NWPP	WECC/Northwest Power Pool	Northwest
24- RMRG	WECC/Rockies	Rockies
25- BASN	WECC/Basin	Great Basin

NERC=North American Electric Reliability Corporation, ISO=Independent System Operator NPCC = Northeast Power Coordinating Council, WECC = Western Electricity Coordinating Council

* Names are intended to describe approximate locations. Exact regional boundaries do not necessarily correspond to state borders or to other regional naming conventions.

Source: U.S. Energy Information Administration.

Fig. 11: Map of U.S. EIA Electricity Market Module (EMM) regions.

AIA climate zones

AIA_CZ1 AIA Climate Zone 1 , AIA_CZ2 AIA Climate Zone 2 , **AIA_CZ3** AIA Climate Zone 3 , **AIA_CZ4** AIA Climate Zone 4 , **AIA_CZ5** AIA Climate Zone 5 ; all, warm climates, cold climates

The bolded climate zones are categorized as “warm climates;” the remainder are “cold climates.”

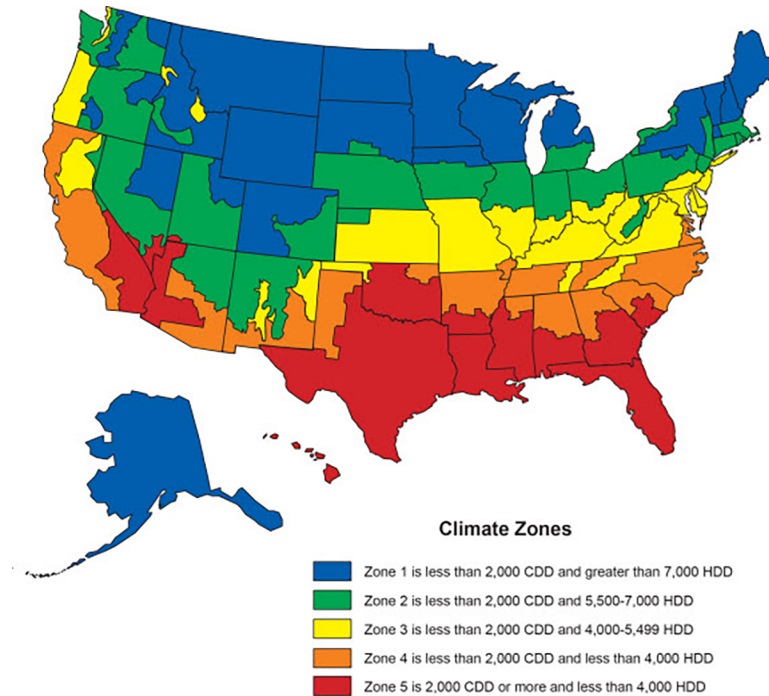


Fig. 12: Map of American Institute of Architects (AIA) climate zones for the contiguous U.S., Alaska, and Hawaii.

Contiguous U.S. states

AL, AZ, AR, CA, CO, CT, DE, DC, FL, GA, ID, IL, IN, IA, KS, KY, LA, ME, MD, MA, MI, MN, MS, MO, MT, NE, NV, NH, NJ, NM, NY, NC, ND, OH, OK, OR, PA, RI, SC, SD, TN, TX, UT, VT, VA, WA, WV, WI, WY; all, warm climates, cold climates

The bolded states are categorized as “warm climates;” the remainder are “cold climates.”

Building type

Residential: single family home, multi family home, mobile home; all residential

Commercial: assembly, education, food sales, food service, health care, lodging, large office, small office, mercantile/service, warehouse, other, unspecified; all commercial

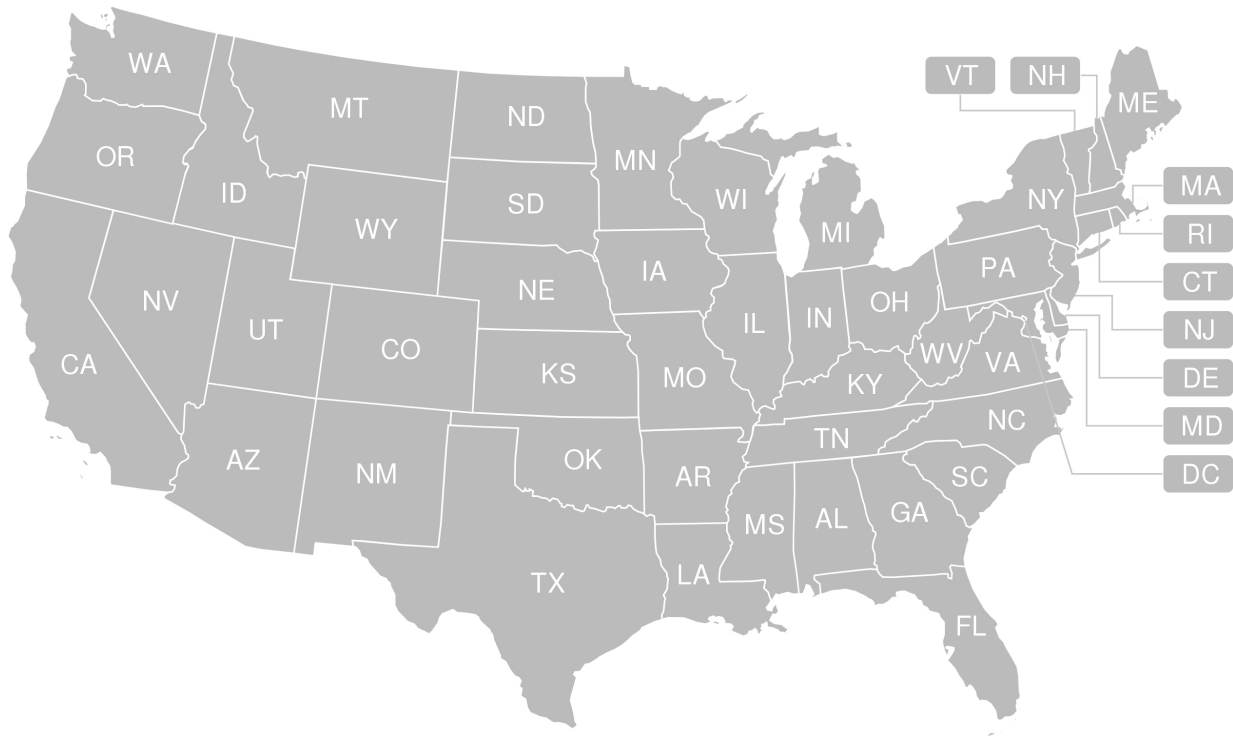


Fig. 13: Map of contiguous United States.

Note

“all” can be used instead of specifying both “all residential” and “all commercial” if all residential *and* commercial building types apply.

Structure type

new, existing; all

Fuel type

Residential: electricity, natural gas, distillate, other fuel; all

Commercial: electricity, natural gas, distillate; all

End use

The end use names appear verbatim in the first column of the tables for residential and commercial buildings.

Note

While “all” is available for specifying all of the end uses in residential and/or commercial buildings (depending on the building types selected), its use should be limited to ECMs where a single technology can credibly affect all energy use in the building. Using the “all” option for end uses also significantly increases computational expense, and that expense will scale exponentially if uncertainty is present on any of the ECMs in the analysis.

Residential

1.8. ECM Definition	End Use	Fuel Type			
		electricity	natural gas	distillate	other fuel
	heating	X	X	X	X
	secondary heating	X	X	X	X

End Use	Fuel Type		
	electricity	natural gas	distillate
heating	X	X	X
cooling	X	X	
ventilation	X		
water heating	X	X	X
lighting	X		
refrigeration	X		
cooking	X	X	
PCs *	X		
non-PC office equipment *	X		
MELs *	X		
other *		X	X
unspecified *	X	X	X
all	X	X	X

* These end uses and all associated technologies may currently only be specified for the *add-on measure type* due to the lack of available baseline cost, performance, and lifetime data for the associated technologies.

Technology

Technology names appear verbatim. For residential building types, the lighting technology names are in the body of the table, categorized by illumination technology (e.g., incandescent, fluorescent) and application or fixture type. For commercial building types, the lighting technology names are categorized generally by bulb type or application. In both cases, these categories are provided for convenience and are not used anywhere in an ECM definition.

Tip

If the technology name for a given end use and fuel type is indicated as null, the ECM definition should have the *unquoted* text “null” written into the *technology* field.

Note

“all” is available as an option to specify all of the technology names that apply to all of the building types, fuel types, and end uses specified for the applicable baseline market. In addition, “all” can be made specific to a particular end use by specifying “all” followed by the end use name – “all heating” or “all water heating,” for example. This shorthand will capture all of the technologies in the named end use that apply to the building types and fuel types included in the applicable baseline market. For example, if the building type is “single family homes” and the fuel type is specified as [“electricity”, “natural gas”] then “all heating” will include all of the heating technologies for residential buildings that use electricity or natural gas.

Residential – Supply

- heating
 - electricity: ASHP air-source heat pump , GSHP ground-source heat pump , resistance heat
 - natural gas: NGHP air-source natural gas heat pump , boiler (NG), furnace (NG)
 - distillate: boiler (distillate), furnace (distillate)
 - other fuel: furnace (LPG), furnace (kerosene), stove (wood)

- secondary heating
 - electricity: secondary heater
 - natural gas: secondary heater
 - distillate: secondary heater
 - other fuel: secondary heater (wood), secondary heater (coal), secondary heater (kerosene), secondary heater (LPG)
- cooling
 - electricity: room AC, ASHP air-source heat pump , GSHP ground-source heat pump , central AC
 - natural gas: NGHP air-source natural gas heat pump
- water heating
 - electricity: electric WH, solar WH
 - natural gas: null
 - distillate: null
 - other fuel: null
- cooking
 - all fuel types: null
- drying
 - all fuel types: null
- lighting

Fixture Type	Bulb Type incandescent/halogen	fluorescent	LED
general service	general service (incandescent)	general service (CFL)	general service (LED)
reflector	reflector (incandescent) reflector (halo-gen)	reflector (CFL)	reflector (LED)
linear fixture		linear fluorescent (T-8) linear fluorescent (T-12)	linear fluorescent (LED)
exterior	external (incandescent) external (high pressure sodium)	external (CFL)	external (LED)

- refrigeration: null
- ceiling fan: null
- fans and pumps: null
- computers: desktop PC, laptop PC, network equipment, monitors
- TVs: home theater and audio, set top box, video game consoles, OTT streaming devices, TV
- other
 - electricity: dishwasher, clothes washing, freezers, rechargeables, coffee maker, dehumidifier, electric other, small kitchen appliances, microwave, smartphones, pool heaters, pool pumps, security system, portable electric spas, smart speakers, tablets, wine coolers
 - natural gas: other appliances

- distillate: other appliances
- other fuel: other appliances

Residential – Demand

roof, wall, infiltration, ground, windows solar, windows conduction, equipment gain, people gain

Commercial – Supply

- heating
 - electricity: `electric_res-heat`, `elec_res-heater` electric resistance heat , `comm_GSHP-heat` commercial ground-source heat pump , `rooftop_ASHP-heat` rooftop air-source heat pump , `elec_boiler` electric boiler , `pkg_terminal_HP-heat` packaged terminal heat pump
 - natural gas: `gas_eng-driven_RTHP-heat` natural gas engine-driven rooftop heat pump , `res_type_gasHP-heat` residential-style natural gas heat pump , `gas_boiler`, `gas_furnace`
 - distillate: `oil_boiler`, `oil_furnace`
- cooling
 - electricity: `rooftop_AC`, `scroll_chiller`, `res_type_central_AC`, `reciprocating_chiller`, `comm_GSHP-cool` commercial ground-source heat pump , `centrifugal_chiller`, `rooftop_ASHP-cool` rooftop air-source heat pump , `wall-window_room_AC`, `screw_chiller`, `pkg_terminal_AC-cool` packaged terminal air conditioner , `pkg_terminal_HP-cool` packaged terminal heat pump
 - natural gas: `gas_eng-driven_RTAC` natural gas engine-driven rooftop AC , `gas_chiller`, `res_type_gasHP-cool` residential-style natural gas heat pump , `gas_eng-driven_RTHP-cool` natural gas engine-driven rooftop heat pump
- ventilation: `CAV_Vent` constant air volume ventilation system , `VAV_Vent` variable air volume ventilation system
- water heating
 - electricity: `solar_water_heater`, `solar_water_heater_north`, `HP_water_heater`, `elec_water_heater`
 - natural gas: `gas_water_heater`, `gas_instantaneous_water_heater`
 - distillate: `oil_water_heater`
- lighting
 - general service: 100W A19 Incandescent, 100W Equivalent A19 Halogen, 100W Equivalent CFL Bare Spiral, 100W Equivalent LED A Lamp,
 - PAR-38: Halogen Infrared Reflector (HIR) PAR38, Halogen PAR38, LED PAR38
 - linear fluorescent: T5 F28, T8 F28, T8 F32, T8 F59, T8 F96
 - low/high bay: T5 4xF54 HO High Bay, Mercury Vapor, Metal Halide, Sodium Vapor
 - other: LED Integrated Luminaire
- refrigeration: Commercial Beverage Merchandisers, Commercial Ice Machines, Commercial Reach-In Freezers, Commercial Reach-In Refrigerators, Commercial Refrigerated Vending Machines, Commercial Supermarket Display Cases, Commercial Walk-In Freezers, Commercial Walk-In Refrigerators
- cooking
 - electricity: `electric_range-combined`
 - natural gas: `gas_range-combined`
- PCs: `null`
- non-PC office equipment: `null`

- MELs: distribution transformers, kitchen ventilation, security systems, lab fridges and freezers, medical imaging, large video boards, coffee brewers, non-road electric vehicles, fume hoods, laundry, elevators, escalators, IT equipment, office UPS, data center UPS, shredders, private branch exchanges, voice-over-IP telecom, point-of-sale systems, warehouse robots, televisions, water services, telecom systems, other
- other: null
- unspecified: null

Commercial – Demand

roof, wall, ground, floor, infiltration, ventilation, windows conduction, windows solar, lighting gain, equipment gain, people gain, other heat gain

1.8.2 Energy efficiency units

Residential – Equipment (Supply)

- Heating
 - Boilers and furnaces (AFUE)
 - Wood stoves (HHV)
 - All other equipment types (COP)
- Secondary heating
 - Electricity (COP)
 - All other fuel types (AFUE)
- Cooling
 - Geothermal heat pumps (EER)
 - All other equipment types (COP)
- Water heating
 - Solar water heaters (SEF)
 - All other water heaters (UEF)
- Refrigeration (kWh/yr)
- Cooking
 - Electricity (kWh/yr)
 - Natural gas (TEff)
 - LPG (TEff)
- Drying (CEF)
- Lighting (lm/W)
- Fans and pumps (kWh/yr)
- Ceiling fan (kWh/yr)
- TVs (kWh/yr)
- Computers (kWh/yr)
- Other
 - Clothes washing (kWh/cycle)

- Dishwasher (kWh/cycle)
- Freezers (kWh/yr)
- Dehumidifier (kWh/yr)
- Microwave (kWh/yr)
- Pool heaters and pumps (kWh/yr)
- Portable electric spas (kWh/yr)
- Wine coolers (kWh/yr)

- All other end uses/equipment types (relative savings (constant) *with add-on measure type* designation)

Commercial – Equipment (Supply)

- Heating (BTU out/BTU in)
- Cooling (BTU out/BTU in)
- Water heating (BTU out/BTU in)
- Ventilation (cfm-hr/BTU in)
- Cooking (BTU out/BTU in)
- Lighting (lm/W)
- Refrigeration (BTU out/BTU in)
- PCs (kWh/yr)
- All other end uses/equipment types (relative savings (constant) *with add-on measure type* designation)

Residential and Commercial – Sensors and Controls (Supply)

- All sensors and controls ECMs (relative savings (constant) *or* relative savings (dynamic))

Residential and Commercial – Envelope Components (Demand)

- Windows conduction (R value)
- Windows solar (SHGC)
- Wall, roof, and ground (R value)
- Infiltration
 - Residential (ACH)
 - Commercial (CFM/ft² @ 0.3 in. w.c.)

1.8.3 Installed cost units

Residential – Equipment (Supply)

- All equipment (\$/unit)

Commercial – Equipment (Supply)

- Ventilation (\$/1000 CFM)
- Lighting (\$/1000 lm)
- Heating, cooling, water heating, cooking, and refrigeration (\$/kBtu/h service, e.g., \$/kBtu/h heating)
- All other equipment (\$/ft² floor)

Residential and Commercial – Sensors and Controls (Supply)

- All sensors and controls ECMs (\$/ft² floor)

Residential and Commercial – Envelope Components (Demand)

- Windows (\$/ft² glazing)
- Walls (\$/ft² wall)
- Roof (\$/ft² roof)
- Floor/ground (\$/ft² footprint)

1.8.4 ECM JSON schema

This section outlines the elements of a JSON file that defines an energy conservation measure (ECM) – a technology included for analysis with Scout. More details about ECMs can be found in the [Analysis Approach](#) and [Tutorial 1](#) sections.

Each sub-section corresponds to a single key in the JSON. The details provided for each key include the parent and child fields, valid data types, a brief description of the field, and one or more illustrative examples. Parent keys are above and child keys are below the current key in the hierarchy of a JSON file.

```
{ "parent key": {
  "current key": {
    "child key": "value" }}
```

The data type “none” indicates that null is a valid value for that key. The parent “root” indicates that it is at the top of the hierarchy, that is, there are no parents for that key.

name

- **Parents:** root
- **Children:** none
- **Type:** string

A descriptive name of the technology defined in the ECM. If possible, the name length should be kept to under 55 characters including spaces. The name should not be shared with any other ECM.

```
{ ...
  "name": "Residential Natural Gas HPWH",
  ... }
```

climate_zone

- **Parents:** root
- **Children:** none
- **Type:** string, list

Either a single climate zone or list of climate zones to which the ECM applies. The climate zone strings must come from the list of [valid entries](#) in the [ECM Definition Reference](#).

```
{ ...
  "climate_zone": [ "AIA_CZ2", "AIA_CZ3", ... ],
  ... }
```

```
{...
  "climate_zone": ["ERCT", "CAMX", "RMPA", "AZNM", "NEWE", "NWPP", ...],
  ...}
```

```
{...
  "climate_zone": ["AL", "AZ", "AR", "CA", "CO", "CT", "DE", "DC", "FL", ...],
  ...}
```

bldg_type

- **Parents:** root
- **Children:** none
- **Type:** string, list

A single building type or a list of residential and/or commercial building types in which the ECM could be installed. The building types specified must be from the list of *valid entries* in the *ECM Definition Reference*.

```
{...
  "bldg_type": "all residential",
  ...}
```

structure_type

- **Parents:** root
- **Children:** none
- **Type:** string, list

The structure type indicates whether the technology described by the ECM is suitable for application in new construction, completed/existing buildings, or both. *Valid structure types* are *new*, *existing*, or *all*, respectively.

```
{...
  "structure_type": "new",
  ...}
```

Tip

If the ECM technology can be applied to both new construction and existing buildings but with differing energy efficiency, installed costs, and/or service life, those differing values should be specified explicitly in the *energy_efficiency*, *installed_cost*, and/or *product_lifetime* fields. This specification method is explained in the *Detailed input specification* section.

fuel_type

- **Parents:** root
- **Children:** none
- **Type:** string, list

The fuel type(s) should correspond to the energy source(s) used by the technology described in the ECM, and can be specified as a string for a single fuel type or as a list to include multiple fuel types. The fuel type(s) should be drawn from the *list of valid fuel types*.

```
{...
  "fuel_type": "electricity",
  ...}
```

Tip

If the ECM describes a technology that does not use energy directly but affects the energy use of the building, i.e., windows and building envelope, the fuel type should be specified as `all`.

Tip

If *fuel switching* is included in the ECM definition, then the fuel types listed should include all fuel types corresponding to equipment or technologies that can be supplanted by the technology described in the ECM. Further information about using the *fuel_switch_to* field is in the *Technology and/or fuel switching* section.

end_use

- **Parents:** root
- **Children:** none
- **Type:** string, list

The end use corresponds to the type of building function that is served by the technology described in the ECM. The end use can be specified as a single string or, if multiple end uses apply, as a list. The valid end uses depend on the building type(s) and fuel type(s) specified, as indicated in the *end use tables* in the *ECM Definition Reference*.

```
{...
  "end_use": ["heating", "cooling"],
  ...}
```

Tip

If the ECM is describing a technology that affects the heating and cooling load of a building, such as insulation, windows, or an air barrier, the end uses should be given as `["heating", "cooling"]`.

technology

- **Parents:** root
- **Children:** none
- **Type:** string, list

The technology field lists the specific technologies or device types that can be replaced by the technology described by the ECM. A complete listing of *valid technology names* is provided in the *ECM Definition Reference*.

```
{...
  "technology": ["HP water heater", "elec_water_heater", "electric WH"],
  ...}
```

market_entry_year

- **Parents:** root
- **Children:** none
- **Type:** int, none

The market entry year specifies the year that the ECM entered or is expected to enter the market. The year should be given as an integer in the format YYYY. `null` is also an acceptable value for the market entry year, and is interpreted to mean that the ECM is available in the first year simulated in Scout.

```
{...
  "market_entry_year": 2019,
  ...}
```

market_entry_year_source

- **Parents:** root
- **Children:** *notes*, *source_data*
- **Type:** dict, none

The market entry year source indicates the reference from which the market entry year for the ECM was derived. If the market entry year is `null`, the source can also be given as `null` without the dict (see [market_exit_year_source](#)).

```
{...
  "market_entry_year_source": {
    "notes": "",
    "source_data": [{
      "title": "High Efficiency Troffer Performance Specification, Version 5.0",
      "author": "",
      "organization": "U.S. Department of Energy",
      "year": 2015,
      "pages": null,
      "URL": "https://betterbuildingssolutioncenter.energy.gov/sites/default/files/
→ attachments/High%20Efficiency%20Troffer%20Performance%20Specification.pdf"}]],
  ...}
```

market_exit_year

- **Parents:** root
- **Children:** none
- **Type:** int, none

The market exit year indicates the final year that the technology described in the ECM is available for purchase. The year should be formatted as YYYY. `null` is also an acceptable market exit year value, and is interpreted as the technology remaining available through the final year simulated in Scout.

```
{...
  "market_exit_year": null,
  ...}
```

market_exit_year_source

- **Parents:** root
- **Children:** *notes*, *source_data*
- **Type:** dict, none

The market exit year source indicates the original source for the exit year specified for the ECM. The field is formatted identically to the *market_entry_year_source* field. If the market exit year is `null`, the source can also be given as `null` without the dict.

```
{...
  "market_exit_year_source": null,
  ...}
```

energy_efficiency

- **Parents:** root
- **Children:** (optional) values of *climate_zone*, *bldg_type*, *end_use*, *structure_type*
- **Type:** float, dict

The energy efficiency value(s) define the energy performance of the technology being described by the ECM. The numeric values should be given such that they correspond to the required units given in the *energy_efficiency_units* field.

```
{...
  "energy_efficiency": 2.8,
  ...}
```

If it is appropriate for the technology described by the ECM, the energy efficiency can be specified more precisely using one or more of the optional child fields. The values should then be reported in a dict where the keys correspond to the applicable child fields. If multiple levels of specificity are desired, the hierarchy of the nested keys must use the following order: *climate_zone*, *bldg_type*, *end_use* and *structure_type*. Additional information regarding this specification method can be found in the *Detailed input specification* section.

```
{...
  "energy_efficiency": {
    "AIA_CZ1": {
      "heating": 1.05,
      "cooling": 1.3,
      "water heating": 1.25},
    "AIA_CZ2": {
      "heating": 1.15,
      "cooling": 1.26,
      "water heating": 1.31},
    "AIA_CZ3": {
      "heating": 1.3,
      "cooling": 1.21,
      "water heating": 1.4},
    "AIA_CZ4": {
      "heating": 1.4,
      "cooling": 1.16,
      "water heating": 1.57},
    "AIA_CZ5": {
```

(continues on next page)

(continued from previous page)

```

    "heating": 1.4,
    "cooling": 1.07,
    "water heating": 1.7}},
  ...}

```

energy_efficiency_units

- **Parents:** root
- **Children:** (optional) matching *energy_efficiency*
- **Type:** string, dict

This field specifies the units of the reported energy efficiency values for the ECM. The correct energy efficiency units depend on the building type, end use, and in some cases, equipment type of the technology described by the ECM. The units can be determined using the *list of energy efficiency units* in the *ECM Definition Reference*.

```

{...
  "energy_efficiency_units": "COP",
  ...}

```

In cases where the energy efficiency is specified with one or more of the optional keys, if the units are the same for all values, the units can still be reported with a single string. If the units are different for some of the keys used, a dict with a structure parallel to the energy efficiency data should be used to report the units. (Energy efficiency units are not a function of climate zone and do not have to be specified with a climate zone breakdown even if the efficiency varies by climate zone.)

```

{...
  "energy_efficiency_units": {
    "heating": {
      "all residential": "COP",
      "small office": "BTU out/BTU in"},
    "cooling": {
      "all residential": "COP",
      "small office": "BTU out/BTU in"}},
  ...}

```

Energy efficiency can also be specified with relative units, as described in the *Relative energy efficiency units* section, or with probability distributions on some or all values, detailed in the *Probability distributions* section.

energy_efficiency_source

- **Parents:** root
- **Children:** *notes*, *source_data*
- **Type:** dict

This key is used to specify the source of the ECM's energy efficiency (i.e., energy performance) values. The *source_data* field description explains how to specify multiple sources. Any details regarding the relationship between the values in the source(s) and the values in the ECM definition should be supplied in the *notes* field.

```

{...
  "energy_efficiency_source": {
    "notes": "Minimum Luminaire Efficiency value reported in section 1.4, sub-section II.
    ↪ a.2.a.",

```

(continues on next page)

(continued from previous page)

```

"source_data": [{
  "title": "High Efficiency Troffer Performance Specification, Version 5.0",
  "author": "",
  "organization": "U.S. Department of Energy",
  "year": 2015,
  "pages": 5,
  "URL": "https://betterbuildingssolutioncenter.energy.gov/sites/default/files/
→ attachments/High%20Efficiency%20Troffer%20Performance%20Specification.pdf"]}],
...}

```

installed_cost

- **Parents:** root
- **Children:** (optional) values from *bldg_type*, *structure_type*
- **Type:** int, dict

The installed cost field represents the typical total cost of the technology and installation of the technology into a building. Costs should be specified such that they are consistent with the *required units* for the type of technology described by the ECM.

```

{...
  "installed_cost": 14,
  ...}

```

Since installation costs can vary by building type (implicitly by building square footage) and whether the technology is being installed as part of new construction or as a replacement of existing equipment or renovation of an existing building, the costs can be specified in a dict using the indicated optional child fields. The keys should match exactly with the allowable values for each of those fields.

```

{...
  "installed_cost": {
    "all residential": 8,
    "all commercial": 10},
  ...}

```

The installed costs can be specified with detail beyond what is shown using the additional optional child field types, as illustrated for the *energy_efficiency* field. The order of the hierarchy is: *bldg_type*, *structure_type*. Further information about detailed structures for specifying the installed cost is in the *Detailed input specification* section.

cost_units

- **Parents:** root
- **Children:** (optional) matching *installed_cost*
- **Type:** string, dict

Cost units correspond to the installed cost given for the ECM. The cost units should match the *required units* based the type of technology described by the ECM.

```

{...
  "cost_units": "$/1000 lm",
  }

```

If there is only a single cost value, a single units value should be given; if the installed cost is specified by one or more of the optional keys and the various installed costs have different units, the cost units should be specified with the same dict structure as the costs. (Cost units are not a function of climate zone and do not have to be specified with a climate zone breakdown even if the costs vary by climate zone.)

```
{...
  "cost_units": {
    "all residential": "$/unit",
    "all commercial": "$/1000 lm"},
  ...}
```

installed_cost_source

- **Parents:** root
- **Children:** *notes*, *source_data*
- **Type:** dict

This key is used to specify the source of the ECM's installed cost values. The *source_data* field description explains how to specify multiple sources. Any details regarding the relationship between the values in the source(s) and the values in the ECM definition should be supplied in the *notes* field.

```
{...
  "installed_cost_source": {
    "notes": "Table 6.3, average of values reported in Total Installed Cost column for
↳ the Gas Storage water heater equipment type.",
    "source_data": [{
      "title": "Energy Savings Potential and RD&D Opportunities for Commercial Building
↳ Appliances (2015 Update)",
      "author": "Navigant Consulting; William Goetzler, Matt Guernsey, Kevin Foley, Jim
↳ Young, Greg Chung",
      "organization": "U.S. Department of Energy",
      "year": 2016,
      "pages": 80,
      "URL": "http://energy.gov/sites/prod/files/2016/06/f32/DOE-BT0%20Comm1%20Appl
↳ %20Report%20-%20Full%20Report_0.pdf"}]],
  ...}
```

product_lifetime

- **Parents:** root
- **Children:** (optional) values from *bldg_type*
- **Type:** int, dict

The product lifetime is the expected usable life of the technology described by the ECM in years. The lifetime value should be an integer greater than 0.

```
{...
  "product_lifetime": 3,
  ...}
```

The product lifetime can be specified by building type, if appropriate for the ECM. The building types are the keys in the lifetime dict and should match the types listed in the *bldg_type* field. Additional information regarding this specification method can be found in the *Detailed input specification* section.

```
{...
  "product_lifetime": {
    "single family home": 10,
    "small office": 7,
    "mercantile/service": 6},
  ...}
```

product_lifetime_units

- **Parents:** root
- **Children:** none
- **Type:** string

The product lifetime units are years. This field is included largely to ensure that the correct units were used when specifying the product lifetime.

```
{...
  "product_lifetime_units": "years",
  ...}
```

product_lifetime_source

- **Parents:** root
- **Children:** *notes*, *source_data*
- **Type:** dict

This key is used to specify the source of the ECM's product lifetime values. The *source_data* field description explains how to specify multiple sources. Any details regarding the relationship between the values in the source and the values in the ECM definition should be supplied in the *notes* field.

```
{...
  "product_lifetime_source": {
    "notes": "Table C-2, Lamp Life column, average of A-Type, Track Lighting, and
↳ Downlights Incandescent Omni rows; converted to years assuming an average use of 8
↳ hours/day.",
    "source_data": [{
      "title": "Energy Savings Forecast for Solid-State Lighting in General Illumination
↳ Applications",
      "author": "Navigant Consulting; Julie Penning, Kelsey Stober, Victor Taylor, Mary
↳ Yamada",
      "organization": "U.S. Department of Energy",
      "year": 2016,
      "pages": 65,
      "URL": "http://energy.gov/sites/prod/files/2016/09/f33/energysavingsforecast16_2.
↳ pdf"}]},
  ...}
```

tsv_features

- **Parents:** root
- **Children:** *shed*, *shift*, *shape*, (optional) values of *climate_zone*, *bldg_type*, and *end_use*
- **Type:** dict

This key specifies the time-sensitive (e.g., hourly or sub-annual) impacts of the technology being described by the ECM. One or more time sensitive ECM features may be described, including *shed*, *shift*, and/or *shape*. Each feature is indicated as a dict key as follows.

```
{...
  "tsv_features": {
    "shed": {...}},
  ...}
```

If an ECM has multiple time sensitive features, they may be specified as follows.

```
{...
  "tsv_features": {
    "shed": {...},
    ...,
    "shape": {...}},
  ...}
```

Optionally, a user may break out time sensitive features by region, building type, and/or end use by setting these variables as the first levels in the dict key hierarchy, followed by the time sensitive feature type key.

```
{...
  "tsv_features": {
    <region 1> : {
      <building type 1> : {
        <end use 1>: {
          <time sensitive feature>: {<feature details>}}}}, ...
    <region N> : {
      <building type N> : {
        <end use N>: {
          <time sensitive feature>: {<feature details>}}}},
  ...}
```

Note that if a region, building type, and/or end use breakout is used, keys for *all* the ECM's applicable regions, building types, and/or end uses must be included. For example, if the time sensitive features dict is broken out by end use, and the ECM applies to both heating and cooling, *both* the heating and cooling keys must be reflected in the time sensitive features dict.

tsv_source

- **Parents:** root
- **Children:** *notes*, *source_data*, (optional) *climate_zone*, *bldg_type*, *end_use*, *shed*, *shift*, *shape*
- **Type:** dict

This key is used to specify the source of the ECM's time sensitive valuation data. The *source_data* field description explains how to specify multiple sources. Any details regarding the relationship between the values in the source(s) and the values in the ECM definition should be supplied in the *notes* field.

```
{...
  "tsv_source": {
    "notes": "Study provides estimate of commercial load curtailment magnitude.",
    "source_data": [{
      "title": "Characterization of demand response in the commercial, industrial, and
↪residential sectors in the United States",
      "author": "Sila Kiliccote, Daniel Olsen, Michael D. Sohn, Mary Ann Piette",
      "organization": "Lawrence Berkeley National Laboratory",
      "year": 2015,
      "pages": 17,
      "URL": "https://onlinelibrary.wiley.com/doi/abs/10.1002/wene.176"}]},
  ...}
```

Optionally, a user may break out time sensitive source data by region, building type, and/or end use by setting these variables as the first levels in the dict key hierarchy, followed by the time sensitive feature type key.

```
{...
  "tsv_source": {
    <region 1> : {
      <building type 1> : {
        <end use 1>: {
          <time sensitive feature>: {
            "notes": <notes>,
            "source_data": [{
              "title": <title>,
              "author": <author>,
              "organization": <organization>,
              "year": <year>,
              "pages": [<start page>, <end page>],
              "URL": <URL>}]}}}}, ...
    <region N> : {
      <building type N> : {
        <end use N>: {
          <time sensitive feature>: {
            "notes": <notes>,
            "source_data": [{
              "title": <title>,
              "author": <author>,
              "organization": <organization>,
              "year": <year>,
              "pages": [<start page>, <end page>],
              "URL": <URL>}]}}}},
    ...}
```

Note that if a region, building type, and/or end use breakout is used, keys for *all* the ECM's applicable regions, building types, and/or end uses must be included. For example, if the time sensitive features dict is broken out by end use, and the ECM applies to both heating and cooling, *both* the heating and cooling keys must be reflected in the time sensitive features dict.

measure_type

- **Parents:** root
- **Children:** none
- **Type:** string

This field is used to specify whether the technology described by the ECM could be substituted for a component already installed in buildings, such as an electric cold-climate heat pump being substituted for an electric furnace and central AC system, or enhance the efficiency of an existing component, such as a window film applied to an existing window or an HVAC controls system that improves the efficiency of existing HVAC equipment. The measure type is then either "full service" or "add-on", respectively. Supplementary information and illustrative examples of the use of this field are available in the [Add-on type ECMs](#) section.

```
{...
  "measure_type": "full service",
  ...}
```

ref_analogue

- **Parents:** root
- **Children:** none
- **Type:** boolean

This field is used to specify whether a reference case analogue copy of the ECM (with the same baseline markets as the ECM, but reference case performance, cost, and lifetime characteristics) should be prepared for subsequent competition with the ECM (and all other ECMs with overlapping baseline markets). Excluding this attribute or setting it to false will prevent preparation of the analogue copy.

```
{...
  "ref_analogue": true,
  ...}
```

fuel_switch_to

- **Parents:** root
- **Children:** none
- **Type:** string

If the ECM replaces a comparable baseline technology or technologies that are served by a different fuel type, this field should identify the fuel type that the ECM switches to. The switched to fuel type name should match exactly with one of the *fuel types* listed in the *ECM Definition Reference*. If the ECM fuel type matches that of the comparable baseline technology, this field can be given as null. Additional information regarding the use of this field is available in the *Technology and/or fuel switching* section.

```
{...
  "fuel_switch_to": "natural gas",
  ...}
```

tech_switch_to

- **Parents:** root
- **Children:** none
- **Type:** string

If the ECM technology differs from that of the comparable baseline technology it replaces, this field should identify the technology that the ECM switches to. The switched to technology name should match one of those shown in the table in [Table 1](#). If the ECM technology is a like-for-like replacement of the baseline technology, this field can be given as `null`. Additional information regarding the use of this field is available in the [Technology and/or fuel switching](#) section.

```
{...
  "tech_switch_to": "ASHP",
  ...}
```

market_scaling_fractions

- **Parents:** root
- **Children:** (optional) values from *climate_zone*, *bldg_type*, *end_use*
- **Type:** int, dict, none

The market scaling fraction is used to further reduce the energy use of the applicable baseline market¹ specified for an ECM whose technology corresponds to only a fraction of that market. The market scaling fraction value should be between 0 and 1, representing the desired fraction of the baseline market. If the ECM does not need a market scaling fraction, the field should be given the value `null`.

```
{...
  "market_scaling_fractions": 0.18,
  ...}
```

Market scaling fractions can be separately specified using the optional child fields if relevant to the technology described by the ECM, if the fields are part of the applicable baseline market, and if appropriate source information is provided.

```
{...
  "market_scaling_fractions": {
    "new": 1,
    "existing": 0.43},
  ...}
```

Further information regarding the use of market scaling fractions is in the [Market scaling fractions](#) section.

market_scaling_fractions_source

- **Parents:** root
- **Children:** *title*, *author*, *organization*, *year*, *pages*, *URL*, *fraction_derivation*; none
- **Type:** dict, string, none

The market scaling fractions source identifies the sources that were used to determine the market scaling fraction, including the exact method for deriving the fraction. If the *market_scaling_fractions* field is `null`, the source should also be specified as `null`.

¹ The applicable baseline market is comprised of the *climate_zone*, *bldg_type*, *structure_type*, *fuel_type*, *end_use*, and *technology* fields.

```
{...
  "market_scaling_fractions_source": {
    "title": "Energy Savings Forecast for Solid-State Lighting in General Illumination, Applications",
    "author": "Navigant Consulting; Julie Penning, Kelsey Stober, Victor Taylor, Mary Yamada",
    "organization": "U.S. Department of Energy",
    "year": 2016,
    "pages": 23,
    "URL": "http://energy.gov/sites/prod/files/2016/09/f33/energysavingsforecast16_2.pdf"}
  ,
  "fraction_derivation": "In Figure 4.4, sum of 2015 data for LED - Connected Lighting, LED - Controls, and shed Lighting - Controls."},
  ...}
```

Multiple scaling fraction values can share the same source so long as the calculation procedure for all of the values is provided in the *fraction_derivation* field, however, no more than one source is allowed for each scaling fraction value. If scaling fractions correspond to different sources, the source information can be given in a nested dict with the same top level structure as the scaling fractions themselves. If the market scaling fraction is set to 1 for one of the keys in the nested structure, the source information can be given as a string explaining any assumptions.

```
{...
  "market_scaling_fractions_source": {
    "new": "Assumes that all new commercial buildings are constructed with BAS",
    "existing": {
      "title": "CBECS 2012 - Table B1. Summary table: total and means of floorspace, number of workers, and hours of operation, 2012",
      "author": "U.S. Energy Information Administration (EIA)",
      "organization": "U.S. Energy Information Administration (EIA)",
      "year": "2012",
      "URL": "http://www.eia.gov/consumption/commercial/data/2012/bc/cfm/b1.cfm",
      "fraction_derivation": "37051 ft^2 floor of commercial buildings with BAS / 87093 ft^2 floor total commercial buildings"}},
  ...}
```

retro_rate

- **Parents:** root
- **Children:** none
- **Type:** float, none

This field assigns an ECM-specific retrofit rate to use in stock-and-flow calculations. The retrofit rate value should be specified as a fraction between 0 and 1. For example, 0.1 corresponds to 10% of the existing technology stock being retrofitted annually.

```
{...
  "retro_rate": 0.1,
  ...}
```

retro_rate_source

- **Parents:** root
- **Children:** *notes*, *source_data*
- **Type:** dict

This field is used to specify the source of the ECM's retrofit rate data. The *source_data* field description explains how to specify multiple sources. Any details regarding the relationship between the values in the source and the values in the ECM definition should be supplied in the *notes* field.

```
{...
  "retro_rate_source": {
    "notes": "Increased commercial building retrofit rate to represent the potential
↳ impacts of the DEEP database in accelerating energy savings from commercial retrofits.
↳ ",
    "source_data": [{
      "title": "Accelerating the energy retrofit of commercial buildings using a
↳ database of energy efficiency performance",
      "author": "Sang Hoon Lee, Tianzhen Hong, Mary Ann Piette, Geof Sawaya, Yixing Chen,
↳ Sarah C. Taylor-Lange",
      "organization": "Lawrence Berkeley National Laboratory",
      "year": 2015,
      "pages": 10,
      "URL": "https://eta.lbl.gov/sites/all/files/publications/tianzhen_hong_-_
↳ accelerating_the_energy_retrofit_of_commercial_buildings_using_a_database_of_energy_
↳ efficiency_performance.pdf"]}]}
...}
```

_description

- **Parents:** root
- **Children:** none
- **Type:** string

A one to two sentence description of the ECM. If the ECM is prospective, i.e., describing a technology still being researched, the description should include URLs or other identifying information for additional references that contain further details about the technology.

```
{...
  "_description": "LED troffers for commercial modular dropped ceiling grids that are a
↳ replacement for the entire troffer luminaire for linear fluorescent bulbs, not a
↳ retrofit kit or linear LED bulbs that slot into existing troffers.",
  ...}
```

_notes

- **Parents:** root
- **Children:** none
- **Type:** string

A text field that can be used for explanatory notes regarding the technologies that can be replaced by the ECM, any notable assumptions made in the specification of the ECM, or any other relevant information about the ECM that is not captured by any other field.

```
{...
  "_notes": "Energy efficiency is specified for the luminaire, not the base lamp.",
  ...}
```

_added_by

- **Parents:** root
- **Children:** *name, organization, email, timestamp*
- **Type:** dict

A dict containing basic information about the user that originally created the ECM.

```
{...
  "_added_by": {
    "name": "Maureen Baruch Kilda",
    "organization": "U.S. Department of Energy",
    "email": "maureen.b.kilda@hq.doe.gov",
    "timestamp": "2016-01-28 21:17:35 UTC"
  }
  ...}
```

_updated_by

- **Parents:** root
- **Children:** *name, organization, email, timestamp*
- **Type:** dict

A dict containing basic information that identifies the user that last updated the ECM, identical in structure to the dict in the *_added_by* field. *null* if the ECM has never been modified.

```
{...
  "_updated_by": ``null``
  ...}
```

shed

- **Parents:** *tsv_features*
- **Children:** *relative energy change fraction, start_hour, stop_hour*, (optional) *start_day, stop_day*
- **Type:** dict

This field sheds (reduces) a certain percentage of baseline electricity demand (defined by the parameter *relative energy change fraction*) during certain days of the *reference year* (defined by the parameters *start_day* and *stop_day*) and hours of the day (defined by the parameters *start_hour* and *stop_hour*.)

```
{...
  "shed": {
    "relative energy change fraction": 0.1,
    "start_day": 152, "stop_day": 174,
    "start_hour": 12, "stop_hour": 20
  }
  ...}
```

shift

- **Parents:** *tsv_features*
- **Children:** *relative energy change fraction*, *offset_hrs_earlier*, *start_hour*, *stop_hour*, (optional) *start_day*, *stop_day*
- **Type:** dict

This field shifts baseline energy loads from one time of day to another by redistributing loads reduced during a certain hour range to earlier times of day. The *start_day* and *stop_day* and *start_hour* and *stop_hour* parameters are used to determine the day and hour ranges from which to shift the load reductions, respectively. The magnitude of the load reduction is defined by the *relative energy change fraction* parameter. The *offset_hrs_earlier* parameter is then used to determine which hour range to redistribute the load reductions to.

```
{...
  "shift": {
    "offset_hrs_earlier": 12,
    "relative energy change fraction": 0.1,
    "start_day": 152, "stop_day": 174,
    "start_hour": 12, "stop_hour": 20}
  ...}
```

shape

- **Parents:** *tsv_features*
- **Children:** *custom_daily_savings*, *json-custom-save-ann*, (optional) *start_day*, *stop_day*,
- **Type:** dict

The final type of time sensitive ECM feature applies hourly savings fractions to baseline loads in accordance with a custom savings shape that represents either a typical day or all 8760 hours of the year.

In the first case, custom hourly savings for a typical day are defined in the *custom_daily_savings* parameter; the hourly savings are specified as a list with 24 elements, with each element representing the fraction of hourly baseline load that an ECM saves. These hourly savings are applied for each day of the year in the range defined by the *start_day* and *stop_day* parameters, as for the shed and shift features.

In the second case, the custom savings shape represents hourly load impacts for all 8760 hours in the *reference year*. Here, the measure definition links to a supporting CSV file via the *custom_annual_savings* parameter that is expected to be present in the *.ecm_definitions/energyplus_data/savings_shapes* folder, with one CSV per measure JSON in *.ecm_definitions* that uses this feature.

```
{...
  "shape": {
    "start_day": 152, "stop_day": 174,
    "custom_daily_savings": [
      0.5, 0.5, 0.5, 0.5, 0.5, 0.6, 1, 1.3, 1.4, 1.5, 1.6, 1.8,
      1.9, 2, 1, 0.5, 0.75, 0.75, 0.75, 0.75, 0.5, 0.5, 0.5, 0.5]]
  ...}

{...
  "shape": {
    "start_day": 152, "stop_day": 174,
    "custom_annual_savings": "sample_8760.csv"}
  ...}
```

start_hour

- **Parents:** *shed, shift, shape*
- **Children:** None,
- **Type:** int

This field indicates the hour of the day (from 1 to 24) that application of a time sensitive ECM feature begins.

```
{...  
  "start": 12  
  ...}
```

stop_hour

- **Parents:** *shed, shift, shape*
- **Children:** None,
- **Type:** int

This field indicates the hour of the day (from 1 to 24) that application of a time sensitive ECM feature ends.

```
{...  
  "stop": 20  
  ...}
```

start_day

- **Parents:** *shed, shift, shape*
- **Children:** None,
- **Type:** int, list

This field indicates the day of the year (from 1 to 365) that application of a time sensitive ECM feature begins.

```
{...  
  "start_day": 12  
  ...}
```

The field may alternatively be specified in list format to yield two start day values, which are paired with two *stop_day* values to yield two distinct day ranges of time sensitive feature application.

```
{...  
  "start_day": [1, 335]  
  ...}
```

stop_day

- **Parents:** *shed, shift, shape*
- **Children:** None,
- **Type:** int, list

This field indicates the day of the year (from 1 to 365) that application of a time sensitive ECM feature ends.

```
{...
  "stop_day": 20
...}
```

The field may alternatively be specified in list format to yield two end day values, which are paired with two *start_day* values to yield two distinct day ranges of time sensitive feature application.

```
{...
  "stop_day": [91, 365]
...}
```

relative energy change fraction

- **Parents:** *shed*, *shift*
- **Children:** None,
- **Type:** float

This field indicates fraction of baseline hourly loads that a measure sheds and/or shifts to another time period.

```
{...
  "relative energy change fraction": 0.1
...}
```

offset_hrs_earlier

- **Parents:** *shift*
- **Children:** None,
- **Type:** int

This field indicates the number of hours earlier to shift baseline load reductions.

```
{...
  "offset_hrs_earlier": 12
...}
```

custom_daily_savings

- **Parents:** *shape*
- **Children:** None,
- **Type:** list

This field provides a list of 24 fractions that represent the percentage of baseline load saved in each hour of a typical day.

```
{...
  "custom_daily_savings": [
    0.5, 0.5, 0.5, 0.5, 0.5, 0.6, 1, 1.3, 1.4, 1.5, 1.6, 1.8,
    1.9, 2, 1, 0.5, 0.75, 0.75, 0.75, 0.75, 0.5, 0.5, 0.5, 0.5]
...}
```

custom_annual_savings

- **Parents:** *shape*
- **Children:** None,
- **Type:** string

This field points to a CSV file containing measure savings fractions for all 8760 hours of the year.

```
{...
"custom_annual_savings": "sample_8760.csv"
...}
```

notes

- **Parents:** *market_entry_year_source, market_exit_year_source, energy_efficiency_source, installed_cost_source, product_lifetime_source, tsv_source, retro_rate_source*
- **Children:** none
- **Type:** string

The notes field should include the exact location of the specific information used from the source(s) identified. The location information should include the table or figure number, and if the value is drawn from tabular data, the applicable row and column heading(s). The notes should also outline any calculations performed to convert from the values found in the source(s) to the value used in the ECM definition, including unit conversions and methods for combining multiple values (e.g., averaging, market share-weighted averaging). Any other assumptions regarding the derivation of the related value in the ECM definition should also be included.

```
{...
"notes": "Value drawn from Table 1 for the Ventless or Vented Electric, Standard_
↪product type. For clothes drying, the expected units of EF (Energy Factor) are_
↪equivalent to lbs/kWh.",
...}
```

fraction_derivation

- **Parents:** *market_scaling_fractions_source*
- **Children:** none
- **Type:** string

For the market scaling fractions, this field should provide a description of how the values were calculated. The description should have enough detail for another user to be able to easily repeat the calculations.

```
{...
"fraction_derivation": "Sum of 2015 data for LED - Connected Lighting, LED - Controls,
↪and shed Lighting - Controls.",
...}
```

source_data

- **Parents:** *market_entry_year_source, market_exit_year_source, energy_efficiency_source, installed_cost_source, product_lifetime_source, tsv_source, retro_rate_source*
- **Children:** *title, author, organization, year, pages, URL*
- **Type:** list

A list that encloses one or more dicts, where each dict corresponds to a single source and includes all of the child fields listed.

```
{...
  "source_data": [{
    "title": "ENERGY STAR Program Requirements: Product Specification for Clothes Dryers",
    "author": null,
    "organization": "U.S. Environmental Protection Agency",
    "year": 2014,
    "pages": "2-3",
    "URL": "https://www.energystar.gov/sites/default/files/specs//ENERGY%20STAR%20Final
    %20Version%201%200%20Clothes%20Dryers%20Program%20Requirements.pdf"}],
  ...}
```

title

- **Parents:** *source_data, market_scaling_fractions_source*
- **Children:** none
- **Type:** string

The title of the source document.

```
{...
  "title": "ENERGY STAR Program Requirements: Product Specification for Clothes Dryers",
  ...}
```

author

- **Parents:** *source_data, market_scaling_fractions_source*
- **Children:** none
- **Type:** string, none

The names of the author(s) of the publication, if any are identified. If no authors are listed, `null` or an empty string are acceptable values for this field if no authors are identified by name in the source.

```
{...
  "author": null,
  ...}
```

organization

- **Parents:** *source_data, market_scaling_fractions_source*
- **Children:** none
- **Type:** string

The journal publication, organization, or other entity that released the source article, report, specification, test result, or other reference.

```
{...
  "organization": "U.S. Environmental Protection Agency",
  ...}
```

year

- **Parents:** *source_data, market_scaling_fractions_source*
- **Children:** none
- **Type:** int

The year that the source was published or last updated.

```
{...
  "year": 2014,
  ...}
```

pages

- **Parents:** *source_data, market_scaling_fractions_source*
- **Children:** none
- **Type:** int, string, none

The page number(s) of the information in the source document, if applicable. If the source is not divided into pages, this entry can have the value `null`. If the relevant information can be found on a single page, the page number should be given as an integer. If the information is divided across several pages or a range of pages, the page numbers should be given as a string.

```
{...
  "pages": "24, 26-29, 37",
  ...}
```

URL

- **Parents:** *source_data, market_scaling_fractions_source*
- **Children:** none
- **Type:** string

The URL where the source can be found on the internet. The URL should point directly to the original source file, if possible.

```
{...
  "URL": "https://www.energystar.gov/sites/default/files/specs//ENERGY%20STAR%20Final
  %20Version%201%200%20Clothes%20Dryers%20Program%20Requirements.pdf",
  ...}
```

name

- **Parents:** *_updated_by, _added_by*,
- **Children:** none
- **Type:** string

The name of the author of the initial definition or latest changes to the ECM.

```
{...
  "name": "Maureen Adams",
  ...}
```

organization

- **Parents:** *_updated_by*, *_added_by*,
- **Children:** none
- **Type:** string

The organization or employer with which the *named* author is affiliated.

```
{...
  "organization": "U.S. Department of Energy",
  ....}
```

email

- **Parents:** *_updated_by*, *_added_by*,
- **Children:** none
- **Type:** string

The email address of the *named* author.

```
{...
  "email": "james.clipper@ee.doe.gov",
  ...}
```

timestamp

- **Parents:** *_updated_by*, *_added_by*,
- **Children:** none
- **Type:** string

The date and time at which the relevant changes were completed. The entry should be formatted as YYYY-MM-DD HH:MM:SS, with the time reported in 24-hour [Universal Coordinated Time \(UTC\)](#) if possible.

```
{...
  "timestamp": "2014-03-27 14:36:18 UTC",
  ...}
```

1.9 EIA Updates

Scout's residential and commercial baseline energy use is anchored to the U.S. Energy Information Administration (EIA) Annual Energy Outlook (AEO). When Scout is updated to a new AEO release, we use a small helper script to compare Scout's internal microsegment totals against the official AEO tables.

The script lives in `scout/AEO_update_helpers/baseline_comparison.py`. At a high level it:

- loads Scout's processed AEO microsegments file (`mseg_res_com_cz.json`),
- aggregates energy use by building class, fuel type, and end use,
- queries the EIA AEO API for the matching time-series,
- compares Scout vs. EIA year-by-year, and
- prints summary tables showing any large differences.

1.9.1 Running the EIA update check

1. **Create a local `.env` file (not committed to git)** in the project root (the same folder as `pyproject.toml`) with your EIA API key:

```
EIA_API_KEY=YOUR_REAL_KEY_HERE
```

You can request a free key from EIA at <https://www.eia.gov/odata/register.php>.

2. **Install Scout and its dependencies** into a virtual environment (see *Installation Guide*).
3. **From the project root, run the helper script**, for example:

```
python -m scout.AEO_update_helpers.baseline_comparison --year 2025
```

Use `--verbose` to see the underlying API calls and any combinations where data are missing.

The output includes per-combination comparisons and simple roll-up tables by building class and fuel type. These reports make it easier to confirm that Scout's baseline matches the chosen AEO reference.