Resolve

Energy & Environmental Economics, Inc.

Jun 25, 2025

CONTENTS

1 `Resolve` Data Flow **3**

2 Contact Us **5**

2.1 Kicking it off 5

Welcome to the new `Resolve` documentation! These docs are periodically updated and improved for better user experience.

`Resolve` is a least-cost capacity expansion model that identifies optimal electricity supply portfolios through capacity expansion and production simulation modeling. Optimal investment plans account for the capital costs of new resources, the variable costs of reliably operating the grid, and additional values such as environmental attributes. `Resolve` has been used to support resource planning and valuation for dozens of clients across North America.

More detailed information along with relevant projects and reports that have leaned on RESOLVE can be found on this webpage.

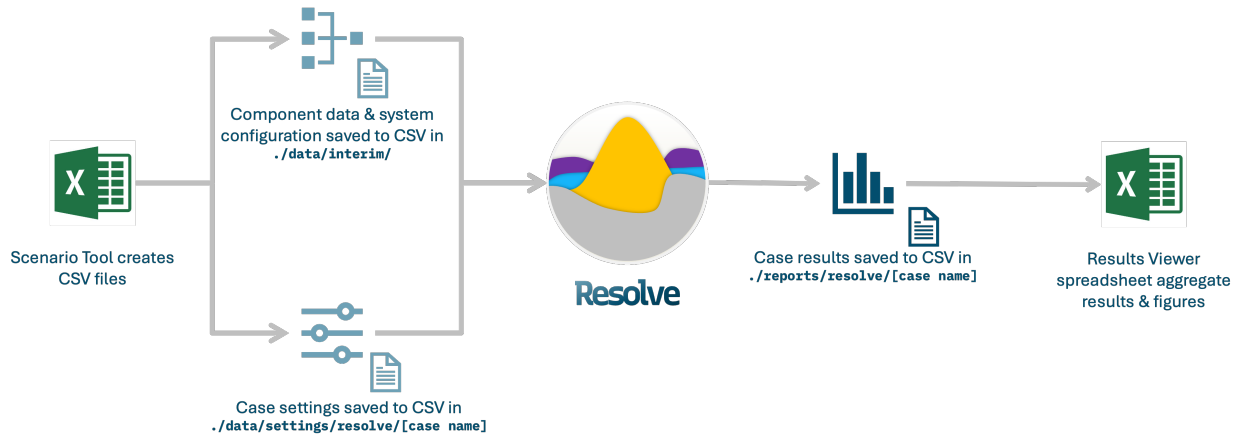
`Resolve` was initially developed in 2014 to assess the investment needs of systems seeking to integrate large quantities of variable renewable resources. In 2020, `Resolve` was re-developed from the ground up to study emerging questions around emerging technologies (e.g., long-duration energy storage, electrolytic fuel production) thanks to funding by the California Energy Commission (CEC EPC-19-056).

CHAPTER

ONE

RESOLVE DATA FLOW

`Resolve` uses a combination of Excel spreadsheet, Jupyter notebooks, and Python scripts. This documentation will briefly cover each step.



CHAPTER

TWO

CONTACT US

Please file bug reports on GitHub.

For support from E3 staff related to the 2022-23 CPUC IRP proceeding, use this booking link

For any other questions or more information on `Resolve` and E3's other modeling tools, email us: platform@ethree.com.

2.1 Kicking it off

Now that we have some background on what `Resolve` is, let's understand and walk through the steps of running `Resolve` and plan clean and reliable electric systems.

2.1.1 Getting Started & Installation

This version of `Resolve` requires Python 3 and either Office 365 Excel, Excel 2021, or later. This page goes through instructions to set up Resolve on your local computer.

System Requirements

- Supported Operating Systems:
  - Windows: Has been tested on Windows 10, Windows 11, and Windows Server 2022
  - macOS: Has been tested on macOS Big Sur (macOS 11) and above.
  - Linux: Has been run on Ubuntu, but other distributions may work. Notably, Excel Scenario Tool does **not** work (since Excel is not available on Linux)
- Python: 3.9+ (via Anaconda distribution)
- Excel: Excel for Microsoft 365, Excel 2021, or later

Python Installation

The computational logic, optimization and associated code for `RESOLVE` is built in the `Python` programming language. Installing the right version and the related dependencies is very important to make sure that the code and the model runs as intended.

E3 recommends downloading the Anaconda distribution package as this is an open source and widely used distribution platform for working in python.

Depending on your operating system and local computer configurations, detailed instructions for set-up and installation can be found on this website

It is important to note there are other available platforms as well - as long as you have Python 3.9 + installed, you should be good.

Github

- `Resolve` uses a combination of Excel spreadsheets, Jupyter notebooks, and Python scripts.
- Github helps in maintaining, operating and structuring these file types so that the user can work with the contents of the model seamlessly.
- The latest release of `Resolve` can be downloaded from GitHub
- You can use a python software distribution like PyCharm (Recommended by E3) that enables you to clone the repository.
- What is Cloning?
  - Cloning in GitHub means making a copy of a repository (a project or codebase) from GitHub to your local computer.
  - Cloning is especially helpful as it helps keep track of changes to files/folders, maintain version control and revert back to original files if need be.

2023 CPUC IRP

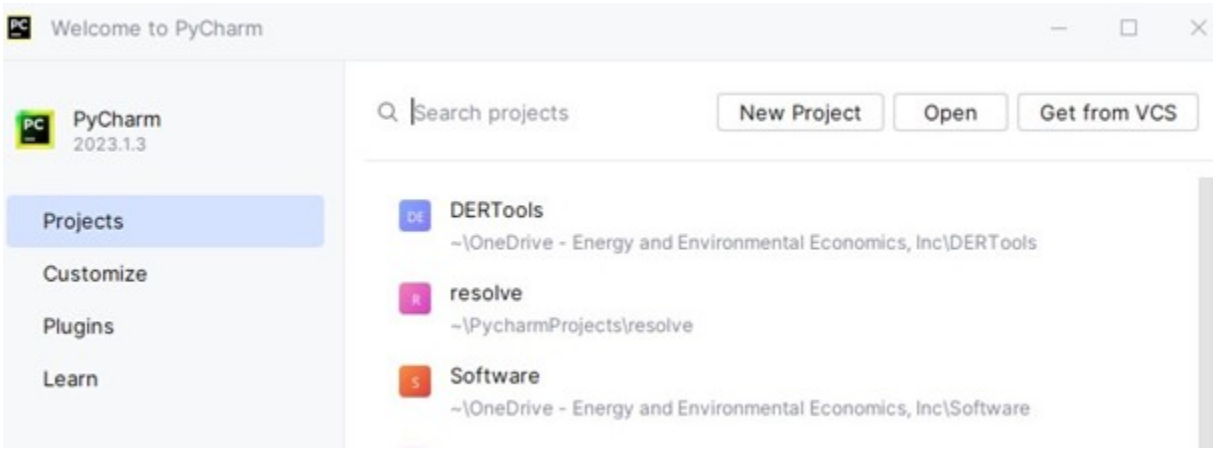
Stakeholders for the 2023 California Public Utilities Integrated Resource Planning (2023 CPUC IRP) process can download `Resolve` and additional data, ruling case results directly from the 2022-23 IRP Events & Materials page.

Pycharm

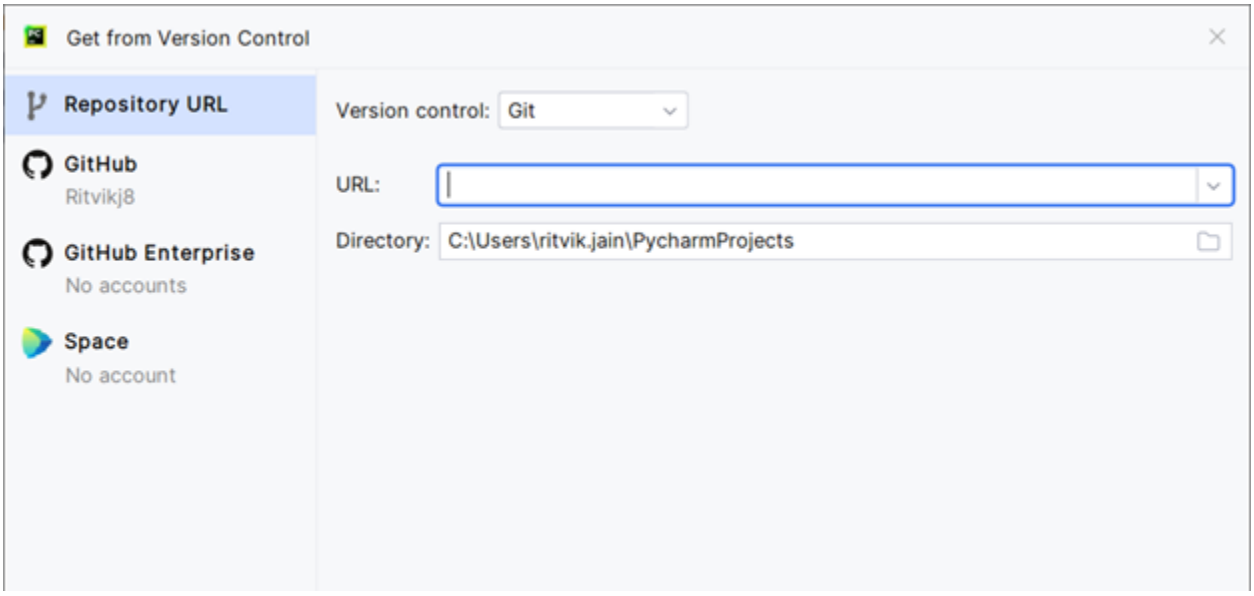
If you are a user who plans to use the package repeatedly and would like to have version control set-up for different releases downloading and setting up Pycharm is highly recommended. One-time installation instructions for Pycharm can be found here

Once you have Pycharm installed, follow these steps in order to clone the `Resolve Repository`

Step 1: Open Pycharm and in the main menu toggle over to projects and click on *Get from VCS*

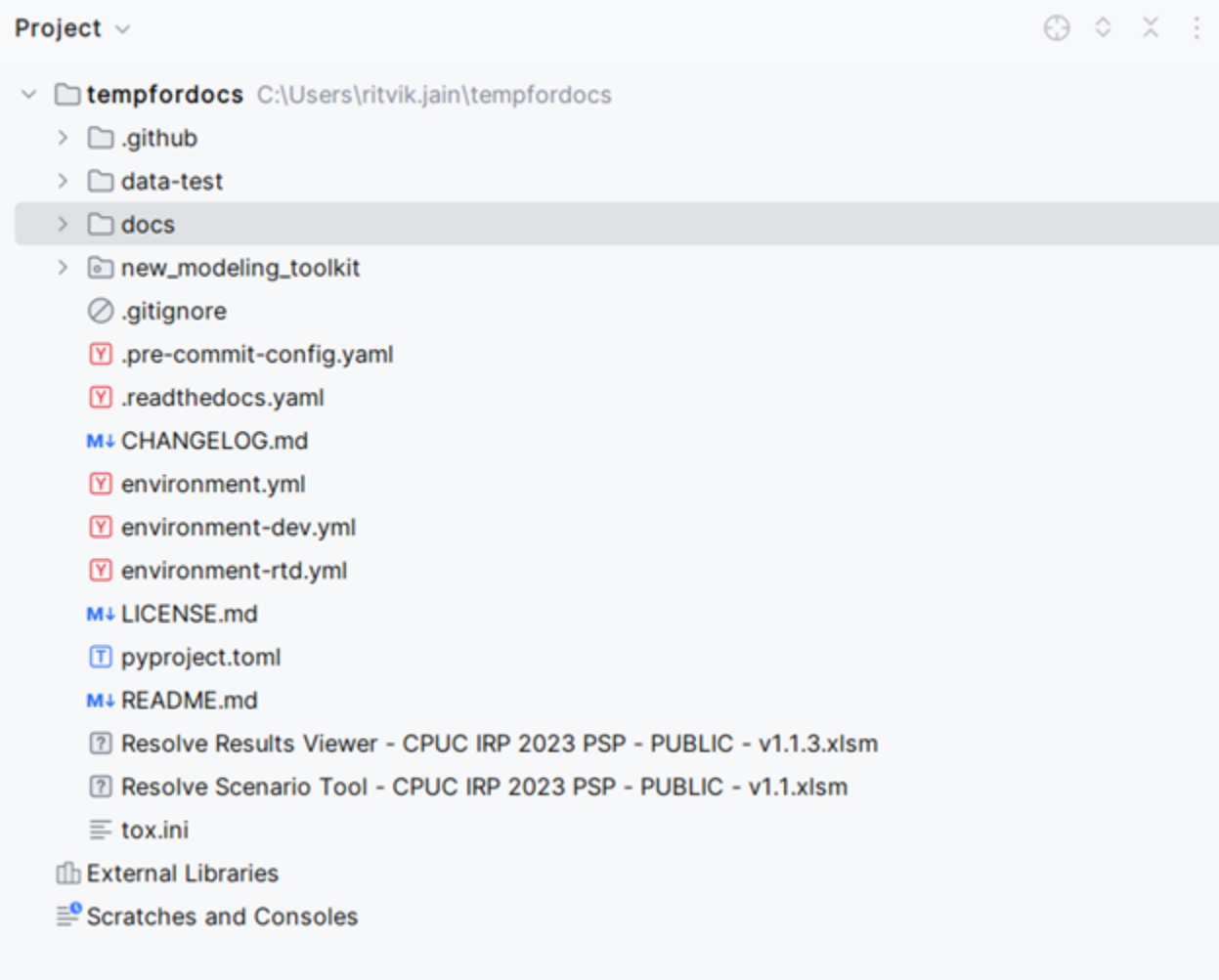


Step 2: After completing step 1, a new window should pop up as shown below. Click on 'Repository url'. make sure version control is set to 'Git', define the directory on your local computer where you would like the `Resolve` model to be saved (Recommended free space on loacl disk is ~10GiB) In the URL section copy and paste the latest release of `Resolve` - which can be found here

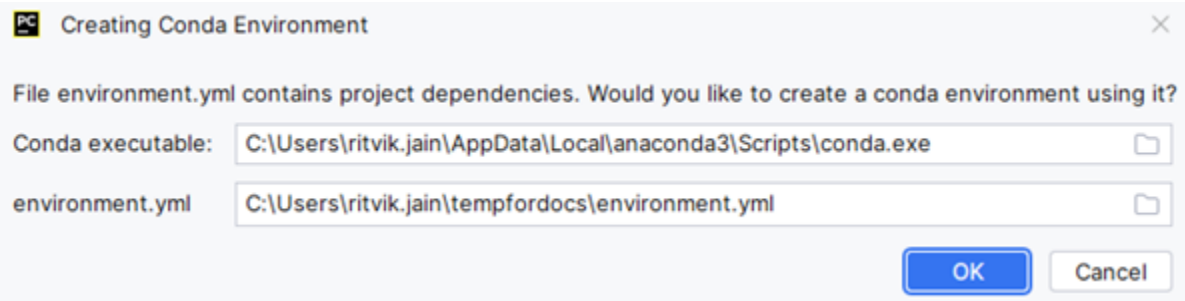


Step 3: Click on Clone at the bottom of the window and allow a couple minutes for Github to clone the repository. The user should be able to see a small dialog box at the bottom of the Pycharm window that reads 'Cloning source repository'

Once this is completed, you should be able to see the necessary files and folders both within pycharm and in your selected local directory. Which should look like the following.



There is a possibility that Pycharm would ask you to create an environment with a pop-up that looks as follows. We will get to this part later so the user can click 'Cancel' and move ahead for now.



Tip: If you have downloaded the resolve package from the CPUC website and or downloaded the zipfile package, you can still go into pycharm - go into the Menubar -> Files -> Open -> Select the folder where you downloaded unzipped resolve and click OK. This enables you to work with existing files and/or circumvent the cloning process.

Files & Data Structure

Once you've downloaded or cloned the Resolve package, you should see (at a minimum) the following files & sub-folders:

- **LICENSE.md:** GNU AGPLv3 open-source license used for `Resolve`
  - **data/:** Data folder for any pre-existing data & case settings
  - **src/resolve/:** `Resolve` source code
- **environment.yml:** Python environment settings
- **pyproject.toml:** Python dependencies
- **User Guide** .pdf
- **Scenario Tool** .xlsm
- **Results Viewer** .xlsm

2023 CPUC IRP Files

The 2023 CPUC IRP release will also include additional files or files that have been renamed:

- **CPUC IRP Resource Cost & Build - PUBLIC** .xlsx
- **Scenario Tool** → Resolve Scenario Tool - CPUC IRP 2023 PSP - PUBLIC - v1.1 .xlsm
- **Results Viewer** → Resolve Results Viewer - CPUC IRP 2023 PSP - PUBLIC - v1.1 .xlsm
- **results/:** Results for released 2023-23 PSP cases

2.1.2 Initial Model Set-up

This Section will walk you through on how to install and set up the model. A lot of this should only be one-time so please make sure that this process is carried out carefully.

All about environments

Python environments for this project should be thought of as isolated computational spaces that will have the necessary configurations needed for you to run the specific project or model - which in this case is `Resolve`

Environments help in setting up packages, dependencies, libraries etc by running a simpe command and do not require much computational background and knowledge - thus enabling the usage of the product for a wide array of stakeholders.

There are many softwares that help in creating, activating and maintaining environments - one of which is `Anaconda`

We recommend using the Anaconda Python distribution and package manager. During the installation process, we recommend selecting the "Add Anaconda3 to my `PATH` environment variable" option so that you have easy access to the `conda` command from the command line.

Tip: If you run into any `conda not recognized` or `command not found: conda` messages in the command line in the following steps, this means that you **did not** add Anaconda to your PATH. You can add either rerun the installer (easiest) or manually add Anaconda to your PATH (see these instructions for some help).

Initial conda Set-up

In order for conda to work properly, you will need to initialize your "shell" (command line, e.g., Command Prompt).

Windows

Option 1: Using Command Prompt

If you use Command Prompt, open a new Command Prompt window and enter:

```
conda init cmd.exe
```

Option 2: Using PowerShell

If you use PowerShell, open a new PowerShell window and enter:

```
conda init PowerShell
```

Then, close all PowerShell windows, and open a new PowerShell window using the "Run as Administrator" option (right-click on the PowerShell application icon in the Start Menu to find this option). Then, enter the following command:

```
Set-ExecutionPolicy Unrestricted
```

Then, close the PowerShell window and open a new one.

macOS Terminal

Since macOS Catalina (10.15), the default "shell" program is zsh. These instructions assume you're on a recent version of macOS. Open Terminal and use the following command:

```
conda init zsh
```

Earlier versions of macOS use bash, so replace zsh in the command above with bash.

Creating Environments

Once you have Anaconda set-up and have the necessary Resolve files and folders you are ready to create environments.

We will use the conda command to create an isolated environment for the Resolve to run within, without disturbing any other Python packages you may have already installed (see the conda documentation for more details on conda environments).

To create the conda environment, we will use the environment.yml file at the top level of the repository. Open your shell of choice and navigate into your cloned copy of the repository. Then, run the following command:

- Create an environment called resolve-env:

  ```
  conda env create -f environment.yml
  ```

- In general it is best practice to name your environment for better tractability. That can be as follows:

```
conda env create -f environment.yml --name your-environment-name
```

Environments using Pycharm

It is also possible for users to directly use Pycharm as an interface to create and manage Python based environments for Resolve.

Step 1: Navigate over to your Resolve specific folder on Pycharm - make sure that the file environment.yml is at the top of the directory

Step 2: At the bottom left of your Pycharm Window, click on the terminal icon to open up the terminal. This should show your selected directory

step 3: In the terminal enter the following code

- Create an environment called `resolve-test-env`:

```
conda env create -f environment.yml --name resolve-test-env
```

Note that the name of the environment is non-consequential and can be set per user's preference.

Once you enter the code, it will take about 2-5 minutes for the system to setup your environment - note that this is a one-time exercise.

After the installation is done - the terminal will prompt you to either activate or de-activate the environment, which will look as follows:

```
#
# To activate this environment, use
#
#     $ conda activate resolve-test-env
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) PS C:\Users\ritvik.jain\tempfordocs> conda activate resolve-test-env
(resolve-test-env) PS C:\Users\ritvik.jain\tempfordocs>
```

Activating the environment would mean that you now have the virtual computing capabilities to run the model. If you would like to learn more about virtual environments and dependencies, supplemental information can be found here

Tip: In order to maintain compatability between different versions, and updates of Resolve, it is best practice to use new environments for each new release. Environments do not interact with one another and this would make sure that the user is able to switch back and forth between different resolve versions with ease.

Solvers (optional) for the most part

Optional

The `resolve-env` environment comes with the open-source HiGHS solver, which enables out-of-the-box solving of`Resolve` cases on any platform. Commercial solvers like Gurobi, IBM CPLEX, and FICO XPRESS offer additional solver features & typically substantially faster solve times. If you have licenses for any of these solvers, `Resolve` will work with them; follow the vendor installation & licensing instructions.

[Add point regarding runtime based on solvers]

2.1.3 RESOLVE Scenario Tool

The `Resolve` Scenario Tool is a user-centric model interface, designed to link data, inputs, assumptions, and constraints with the `Resolve` code. The primary step in being able to make that linkage is making sure that xlwings is correctly set-up

Configuring xlwings

![Overall add a figure for describing the flow of inputs into ST from

1. GEnlist
2. RC&B
3. RP&T co-ordinate w ANgineh]

![Additionally we need to add a Data-catalog - table where the data sources are outlined Eg for load we refer to IEPR… To be decided We can add it to the ST but in the doc we can refer back to it]

We have seen that running resolve requires interfacing with both `excel` as well as `python` based code. xlwings is a tool that helps in interfacing excel and python and something that is very necessary for the model to run smoothly

More standalone information on xlwings can be found here:

For our purposes we will focus on installing xlwings and making it work with the Resolve Scenario Tool excel workbook

On the Scenario Tool's `Cover & Configuration` tab, you will need to tell the Scenario Tool how to find your `resolve-env` **Python path** and the **data folder** where you want to save inputs.

Scenario Tool Configuration

User	Roderick (Laptop)
Python Path	*/Users/rgo/mambaforge/envs/resolve-env/bin/python*
Data Folder	*/Users/rgo/PycharmProjects/resolve/data*

Windows Excel

Configure Python Path:

1. Open Command Prompt or PowerShell and activate the environment with the command `conda activate resolve-env`
2. Type the command `where python` to see where the `resolve-env` version of Python is stored. This should look like: `C:\Users\[username]\Anaconda3\envs\resolve-env\python.exe`. Paste this path into the **Python Path** cell.

Warning: Make sure to use a backslash \ (and not a forward slash /) on Windows for the Python path.

Configure Data Folder Path:

By default, this folder is called `data` and located in the project folder next to the `scr` folder. For completeness, type the full path to the data folder, may look something like `~/.../resolve/data`

macOS Excel

Configure Python Path:

1. Open Terminal and activate the environment with the command `conda activate resolve-env`
2. First time setup only Run the command `xlwings runpython install`. You should see a prompt from macOS asking you for permission for Terminal to automate Excel. **You must allow this.**
3. Type the command `which python` to see where the `resolve-env` version of Python is stored. This should look like: `/Users/[username]/anaconda3/envs/resolve-env/bin/python`. Paste this path into the **Python Path** cell.

Warning: Make sure to use a forward slash / (and not a backslash \) on macOS for the Python path.

Configure Data Folder Path:

By default, this folder is called `data` and located in the project folder next to the `src` folder. For completeness, type the full path to the data folder, may look something like `~/.../resolve/data`

Structure & Tabs of the Scenario Tool

Scenario Tool Tab	Short Description
Cover & Configuration	Key starting point, takes input for Python & Folder Path as defined by the user
Case List	A tab where users can look at existing case designs, make new cases and record them
RESOLVE Settings	Primary place to save input data and case settings, includes Macros for running cases(More detail on this tab below)
Temporal Settings & Rep Periods	RESOLVE uses representative days o run the simulatuion, additional details on the same can be found on these tabs
System	A tab to define and create new systems that encapsulate all sectors of the electric systems
Loads	Defines different load components and values for the system
Transmission Paths	Defines Tx paths, forward and reverse ratings as well as hurdle rates

The Scenario Tool has a plethora of other information and tabs that all flow into the model. Not all of these tabs are defined in detail in the documentation, however some of the tabs are self-explanatory.

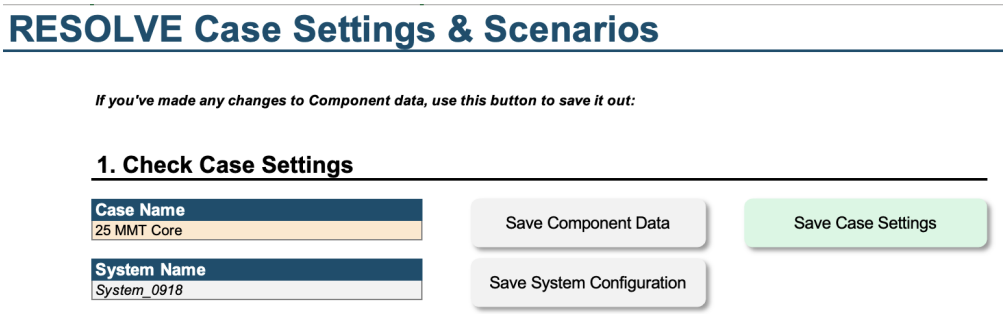
For eg: Different resource types have their own tabs - taking variable resources as an example, the `Variable` tab lists all the variable resources in the CAISO system, along with what zone they are in, what are the operating characteristics of that resource as well as whether or not there is potential to build additional capacity of that resource.

Additional comprehensive information on some of these inputs and assumptions present in the Scenario Tool can be found in the Inputs & Assumptions document released by the CPUC.

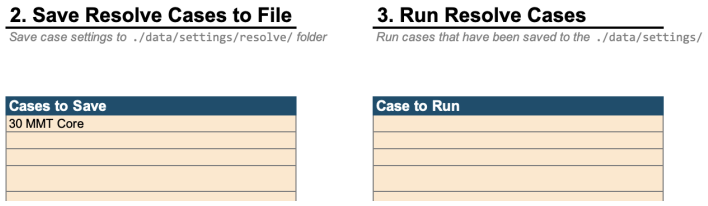
For users planning on comprehensively using the Model, a thorough reading of the I&A document is recommended.

Saving Input Data & Case Settings

Users who want to run pre-existing cases can save all necessary inputs from the "Resolve Settings" tab of the Scenario Tool.



1. Save the `Component` input data (e.g., resource heat rates, etc.) using the "Save Component Data" button
2. Save the `System` configuration (i.e., the combination of loads, resources, etc. to model) using the "Save System Configuration" button
3. Save a single case (e.g., years to model, scenarios, etc.) by selecting a case name from the "Case Name" dropdown and pressing the "Save Case Settings" button
    - To the right side of the "Resolve Settings" tab, you can also save *multiple* cases out of the Scenario Tool:



For users who want to update or create new inputs data, systems, and cases, the subsequent pages discuss how to update data in the Scenario Tool in more detail.

1. `Component` Data

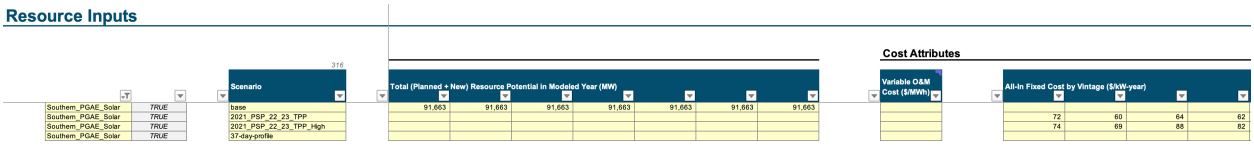
`Components` are the fundamental building block of what `Resolve` models. All `Components` have attributes that can be set via the Scenario Tool.

Class	Description
AnnualEmissionsPolicy	Annual emissions accounting that can encompass generation & importing transmission paths.
AnnualEnergyStandard	Renewable Portfolio Standard (RPS) or Clean Energy Standard (CES)-type policies.
Asset	Any physical asset where we want to track & optimize investment costs.
CandidateFuel	A fuel that can be used by generators (or created if using electrolytic fuel production)
ELCCSurface	ELCC surface inputs
Load	A load component consisting of an hourly profile and an annual energy and/or peak forecast.
PlanningReserveMargin	A Planning Reserve Margin (PRM) reliability accounting constraint, interacts with `ELCCSurface`.
Reserve	Operating reserves, such as spin, regulation, load following.
Resource	An `Asset` used for electric sector operations (e.g., thermal generator, battery, variable resource).
TXPath	`Resolve` uses a "transportation" ("pipe-and-bubble") model for transmission flows between zones.
Zone	A location, constrained by transmission, where loads & resources are located.

Scenario tagging functionality

See input_scenarios for discussion about how to determine which scenario tagged data is used in a specific model run.

On most of the component & linkage attributes tabs, you will find a `Scenario` column. In the Scenario Tool, a single instance of a component can have **multiple** line entries in the corresponding data table as long as each line is tagged with a different scenario tag, as shown in the below screenshot.



Scenario tags can be populated *sparsely*; in other words, every line entry for the same resource does not have to be fully populated across all columns in the data tables. In the example screenshot above, this is demonstrated by the `base` scenario tag having data for "Total (Planned + New) Resource Potential in Modeled Year (MW)" and no data for "All-In Fixed Cost by Vintage ($/kW-year)", whereas the scenario tags `2021_PSP_22_23_TPP` and `2021_PSP_22_23_TPP_High` are the reverse.

The Scenario Tool will automatically create CSVs for all the data entered into the Scenario Tool. These CSVs have a four-column, "long" orientation.

timestamp	attribute	value	scenario (optional)
[None or timestamp (hour beginning)]	[attribute name]	[value]	[scenario name]
…	…	…	…

Timeseries Data

Hourly timeseries data is now stored in separate CSV files under the `./data/profiles/` subfolder to keep the Scenario Tool spreadsheet filesize manageable. These CSVs must have the following format:

timestamp	value
[timestamp (hour beginning)]	[attribute value]
…	…

On the Scenario Tool, you'll see certain data attributes have filepaths as their input, which point the code to the relevant CSV file.

2. Configure a System

The "System" tab defines the energy system being modeled, which is composed of a list of various modeling components (loads, resource, policies, etc.). This tab will have any pre-populated system configurations in the yellow table(s) to the right of the tab, and different systems are linked to the active case on the "`Resolve` Settings" tab.

CPUC System

component	instance
AnnualEmissionsPolicy	CARB Cap-and-Trade
AnnualEmissionsPolicy	GHG Planning Target
AnnualEnergyStandard	RPS
AnnualEnergyStandard	SB 100
Asset	Arizona_Solar_48
Asset	Arizona_Solar_48_Generic
Asset	Arizona_Solar_55
Asset	Arizona_Solar_55_Generic
Asset	Arizona_Solar_62
Asset	Arizona_Solar_62_Generic
Asset	Arizona_Solar_77
Asset	Arizona_Solar_77_Generic
Asset	Arizona_Solar_90

If users add any new modeling components (e.g., load components with new names, resources with new names), they will need to make sure that these new components are added to either an existing or new system configuration by updating the system list.

3. Define Case Settings

This page will discuss the various settings that can be toggled in a `Resolve` run. Most settings on the "Resolve Settings" tab are formula-linked to pre-populated settings on the "Case List" tab. You can update these settings by adding columns to the "Case List" tab.

2023 CPUC IRP

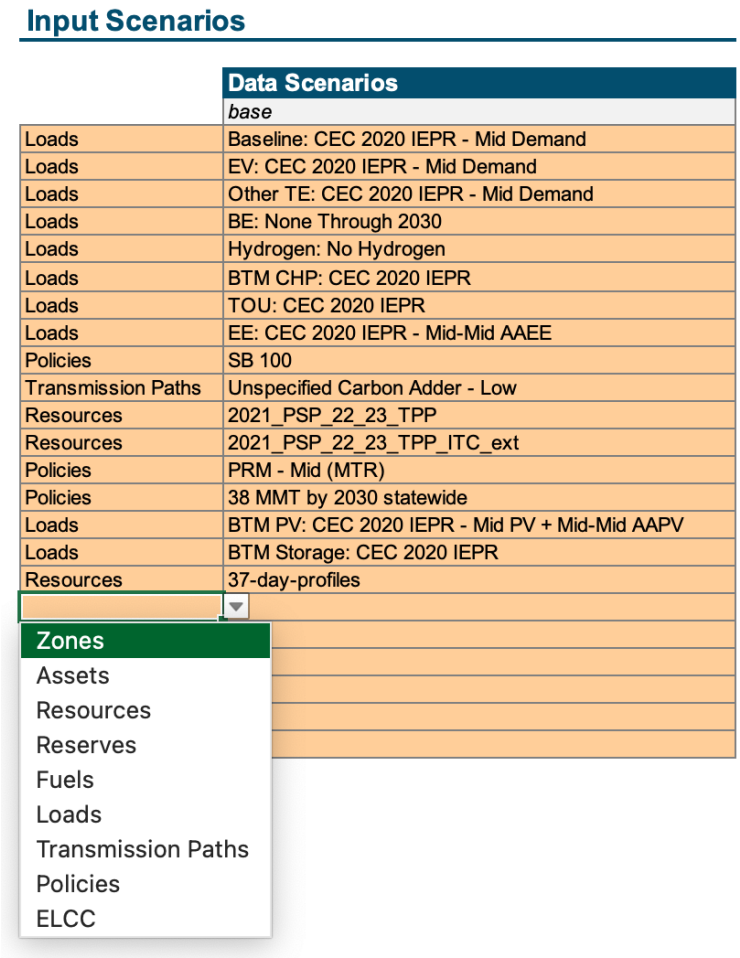
CPUC IRP stakeholders will find case settings for all the cases posted on the CPUC website pre-populated on the "Case List" tab.

Case Settings

Input Scenarios

See scenario_tags for discussion about how to input scenario tagged data for components & linkages.

As discussed in `resolve.common.component.Component.from_csv()`, input scenario tags are prioritized based on the order of scenarios in the `Resolve` case. Scenarios listed toward the bottom of the scenario list are higher priority and more likely to override other data if data is tagged with a "lower priority" scenario tag. In the screenshot below, for example, data tagged with the `base` tag will the lowest priority, since it is the first tag in the scenario list. For any of the subsequent scenario tags (e.g., `2021_PSP_22_23_TPP_ITC_ext`), to the extent that there is data that is tagged with the higher priority scenario tag, that higher priority data will override any `base`-tagged data.



On the `Resolve` Settings tab, users will find an orange dropdown inputs menu to help ensure that input scenarios selected are based on scenario tags that already are defined on the respective component & linkage attribute tabs. In the first column, select the sheet on which to look up available scenario tags. Then, in the second column, the dropdown input should only present scenario tags that are already defined on the respective sheet of the Scenario Too.

Representative Period Settings

Toggle between pre-defined sets of sampled days saved in the Scenario Tool. See {ref}`timeseries-clustering for instructions on how to create new sampled days.

Financial & Temporal Settings

The model will now endogenously calculate the annual discount factors to use for each modeled year based on four pieces of information:

1. **Cost dollar year:** The dollar year that costs are input in & should be reported out in. In general, `Resolve` is designed to be in real dollars for a specific dollar year.
2. **Modeled years:** Which modeled years to include in the `Resolve` case.
3. **End effect years:** The number of years to extrapolate out the last modeled year. In other words, considering 20 years of end effects after 2045 would mean that the last model year's annual discount factor would represent the

discounted cost of 2045-2064, assuming that the 2045 costs are representative of a steady-state future for all end effect years.

4. **Annual discount rate:** Real discount rate in each year
5. **Inter-period dynamics:** Include additional chronological information to allow `Resolve` to shift energy between days across the modeled weather years.

Solver Settings

For now, users must follow the pattern of `solver.[solver name].[solver option].[data type]` when setting the solver settings. For example, users wanting to set Gurobi's `Method` setting would need to enter `solver.gurobi.Method.int` and the corresponding value.

Custom Constraints

Custom constraints allow users to customize the functionality of `Resolve` by adding additional constraints without needing to change the code. These are defined on the "Custom Constraints" tab and saved to `./data/settings/resolve/[case name]/custom_constraints/`

2023 CPUC IRP

For the CPUC IRP, custom constraints are used for various custom functionality:

- Resource deliverability (i.e., CAISO FCDS/EO designation) and accompanying CAISO transmission upgrades
- Connecting disaggregated build variables to aggregate operational resources (which allows `Resolve` to make granular investment decisions while reducing the model size needed to represent operations.
- Group-level constraints (e.g., "Resolve must build 15 GW of offshore wind by 2045" but can select amongst the 4 candidate OSW resources.)

To create custom constraints:

1. Create a "Custom Constraint Group" name of your choosing. These groups are toggled on/off in the active case settings together, so group custom constraints accordingly.
2. Within each custom constraint group, define…
3. To be able to include a set of constraints the user needs to add the name of the custom constraint group to the case settings tab

Timeseries Clustering

Advanced Topic

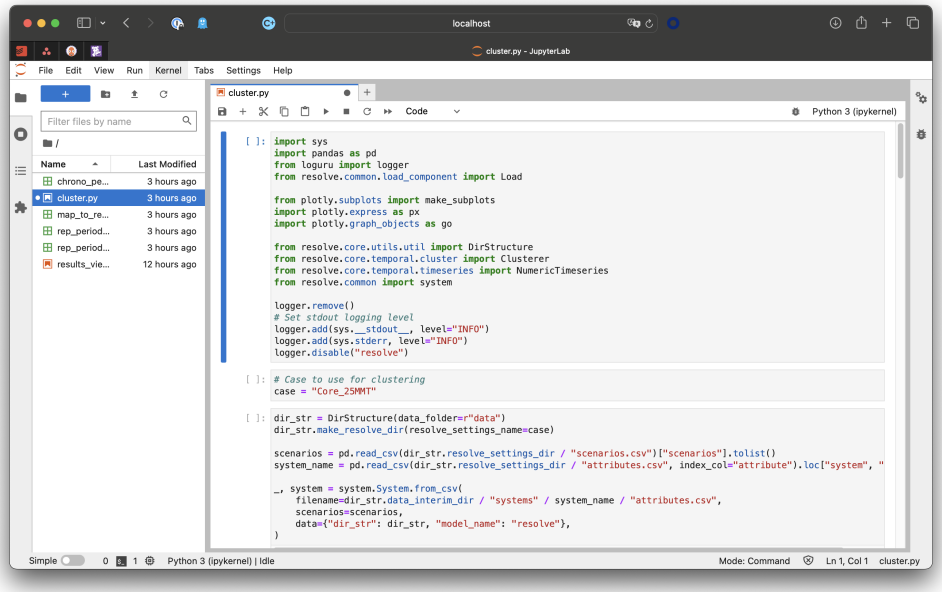
Users who want to create a new set of sampled days can do so using the included Jupyter notebook in `./notebooks/cluster.py`. This is a Jupytext file. Note that to run the timeseries clustering, you first must save a case and all relevant system & component data (as described in saving-inputs).

To open this notebook:

1. Open a Command Prompt or Terminal and navigate to the `./notebooks` subfolder
2. Activate the `resolve-env` environment using `conda activate resolve-env`
3. Open the notebook using the following commands, which will launch a new tab in your web browser:

```
jupytext-config set-default-viewer
jupyter lab cluster.py
```

This will launch a window in your web browser that looks something like this:



4. In the second "cell", change the case name from "Core_25MMT" to the case you want to use. The notebook will then load the case and its corresponding system data to start the timeseries sampling process.
5. Run the cells (using the button or by using `Shift + Enter` keyboard shortcut). (For Jupyter notebook basics, users can start here).
6. If the notebook runs successfully, a CSV file will be created in the `./notebooks` folder called `map_to_rep_periods.csv`. Paste the data in that CSV into the "Rep Periods" tab of the Scenario Tool (insert columns in the yellow input table areas as-needed).
7. Toggle between different timeseries samples by updating your case settings on the "Case List" tab.

2.1.4 Running Resolve

After saving your input data & case settings (refer below to see how this should look like) (as described in saving-inputs), you are now ready to run `Resolve`!

Insert details on input file structure and directories and where reht are saved

Running `Resolve` from the Scenario Tool

As in previous versions of `Resolve`, users can run cases directly from the Scenario Tool. Below the "Run Resolve Cases" header on the right of that tab, you'll find a green "Run Resolve Cases Locally" button.

- On Windows, this will create a new command line window, and you will see `Resolve` progress.
- On macOS, we have not yet figured out how to show the command line window as `Resolve` is running. For now, we recommend macOS users run `Resolve` from Terminal themselves, as described in the next section.

Running Resolve from Command Line

If you plan to run Resolve via a command line/terminal, use the following instructions. Running Resolve via the command line gives you more options for how the model is run than are exposed in the Scenario Tool, as discussed below.

1. In a command line (e.g., Command Prompt), navigate into the ./src/resolve/resolve directory
2. Activate resolve-env conda environment: conda activate resolve-env
3. Use the command python run_opt.py to run a case. The run_opt.py script accepts the following arguments:

- --data-folder: The name of your data folder (if different than the default .\data)
- --solver-name: The name of the solver to use (e.g., gurobi, cplex, amplxpress, appsi_highs)
- --raw-results: Save all raw Pyomo model components as CSVs (for detailed model inspection).
- --symbolic-solver-labels: Enable descriptive variable names in the Pyomo model formulation–helpful for debugging.

Tip: If for the installation process, you had used Pycharm or any other Python software, then the recommended best practice is to run resolve from there after saving the ST as this avoids using macros which might cause computational issues relating to excel based macros as these are deprecated in newer configurations. [Ritvik to rephrase]

Examples:

- Run all cases listed in ./data/settings/resolve/cases_to_run.csv:

```
python run_opt.py
```

- To run a single case called Core_25MMT, type the name of the case into the command line:

```
python run_opt.py Core_25MMT
```

- Run all cases from a different data folder called data-new (listed in ./data-new/settings/resolve/cases_to_run.csv):

```
python run_opt.py --data-folder data-new
```

- Run all cases using cplex as your solver:

```
python run_opt.py --solver-name cplex
```

Note: Hint: If you're in your command line and unsure what arguments to pass to run_opt.py, use the command python run_opt.py --help to get help!

Note for E3 Staff

Instructions for running Resolve on ethree.cloud available on the encyclopedia

2.1.5 RESOLVE Results Viewing

Once `Resolve` is done solving, it will save a series of CSVs that summarize the portfolio investment & operational decisions. These files are stored in the run's report folder, which is found in `./reports/resolve/[case name]/[timestamp]/`

The `Resolve` package includes a spreadsheet Results Viewer, which is powered by VBA and will load the data from the CSVs for viewing as formatted figures & tables. See the spreadsheet for more instructions.

If users pass the `--raw-results` command line argument when running `Resolve`, a CSV for every Pyomo Param, Var, Expression, and Constraint will be created and stored in the run's report folder.

Output File Structures

This section will walk you through what a typical output file structure for RESOLVE looks like

Raw & Summary Results

Some files offer valuable insights compared to others, this section will look at key outputs and summary results of the run

RESOLVE Results Viewer

This section will cover where will the end-of-the-day resolve story live

2.1.6 FAQs

This section will cover FAQs, Common errors, Troubleshooting steps in detail

---

ST FAQ

Tech FAQ

Diagnosing Infeasibilities

Some solvers can help users diagnose infeasibilities in their model. We are aware of that feature in Gurobi, CPLEX, and XPRESS. If `Resolve` runs into an infeasibility and the solver is able to identify the infeasibility, sometimes called Irreducible Infeasible Set (IIS), `Resolve` will save the infeasibility in the run's report folder (e.g., Gurobi will produce a text file with a `.ilp` file extension).

RV FAQ