NREL
NATIONAL RENEWABLE ENERGY LABORATORY

40
1977 – 2017





SAM Webinars 2017: SAM Open Source

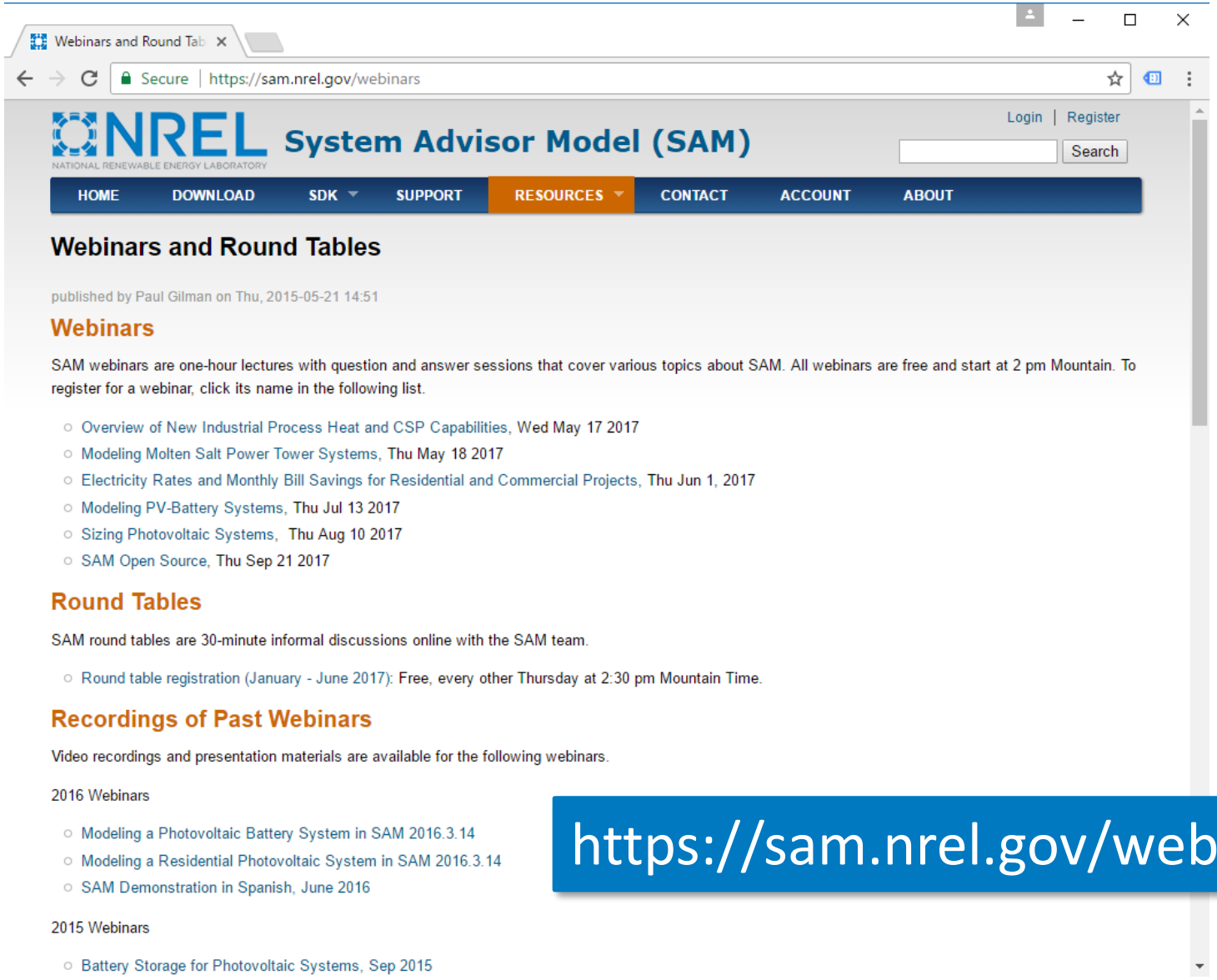
Nicholas DiOrio

September 21, 2017

SAM Webinars 2017

- Overview of New Industrial Process Heat and CSP Capabilities, May 17
- Modeling Molten Salt Power Tower Systems, May 18
- Electricity Rates and Monthly Bill Savings for Residential and Commercial Projects, June 1
- Modeling PV-Battery Systems, July 13
- Sizing Photovoltaic Systems, August 10
- **SAM Open Source, September 21**

Registration Links and Webinar Recordings



https://sam.nrel.gov/webinars

Outline

- Why open-source?
- NREL releases
- Code overview
- License
- Contributing
  - Issues
  - Pull requests
- How do you use public tools?
- Q&A

Open Sourcing SAM

- Increased transparency, flexibility, and collaboration opportunities.
- We are excited to continue working on SAM and fostering a new community of contributors.
- This is the newest in the many ways to interact with SAM, including scripting, the SDK, etc.

What kinds of things can you do with SAM open-source?

Transparency

- Look at the underlying code of a model that you are interested in.

Flexibility

- Change the way a model works for research purposes
- Change electricity rate models to be specific to your country

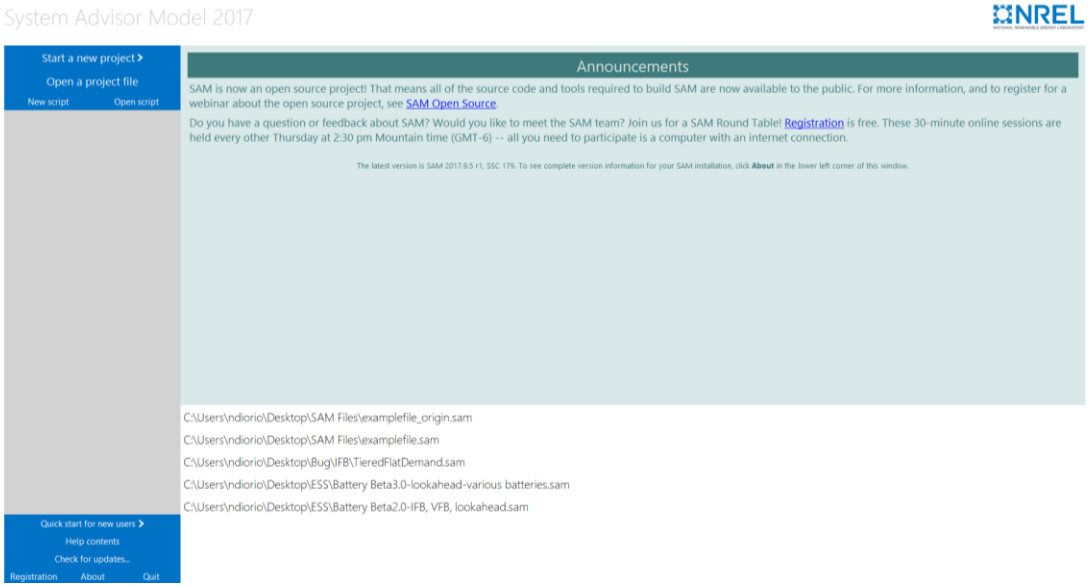
Collaboration

- Add new technology models
- Add a new battery dispatch model

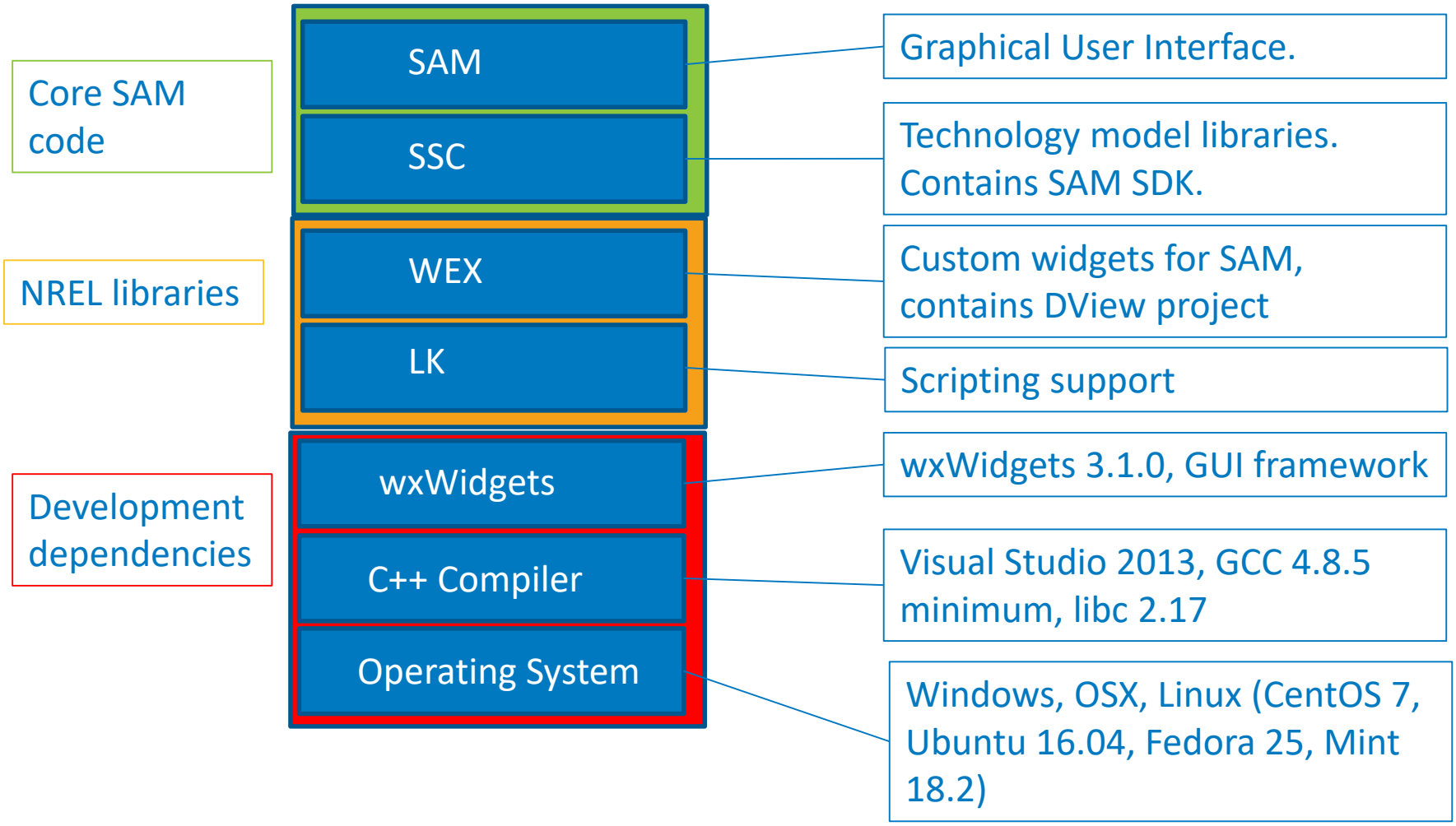
We'd love to learn how you use SAM's open-source code! It helps us tailor our efforts and get funding to develop the tool.

NREL Releases

- NREL will continue to maintain and release official desktop versions of SAM.
  - Releases built from the open-source repositories
  - User contributions can be considered for inclusion in official versions



SAM Code Architecture



SAM Team Resources

- The core SAM team can help get you started in the right area of the code.

Janine Freeman	SAM Project Lead, PV and Wind modeling
Nicholas DiOrio	Open Source Lead, PV and Storage modeling, core SAM code
Nate Blair	Emeritus Project Lead, Financial Modeling
Ty Neises	CSP Models
Mike Wagner	CSP Models
Steve Janzou	Financial Modeling, Core SAM Code
Paul Gilman	Financial Modeling, User Support

Code Locations

wxWidgets	https://www.wxwidgets.org/downloads/
LK	https://github.com/NREL/lk
WEX	https://github.com/NREL/wex
SSC	https://github.com/NREL/ssc
SAM	https://github.com/NREL/SAM

If you are new to Git and GitHub, please checkout:
https://guides.github.com/

Code licenses (LK and WEX)

- Licensed under an MIT-type license. Main restrictions:
  - Redistribution of source code or binary must reproduce copyright notice, list of license conditions, and disclaimer.
  - Neither the name of the copyright holder or the names of contributors may be used to endorse products derived from the software without prior written permission.
- See full licenses:
  https://github.com/NREL/lk/blob/develop/LICENSE.md
  https://github.com/NREL/wex/blob/develop/LICENSE.md

Code licenses (SSC and SAM)

- Licensed under a mixed MIT-type license and GPLv3 license.
- Commercial businesses can use SSC and SAM under the MIT-type restrictions
  - You can use SSC and SAM in software you develop for your business.
- Research entities, including national labs, institutions of higher learning, and non-profits are restricted under a GPLv3-type license.
  - You can use SSC and SAM in your research, but must make your changes publicly available.

Code licenses (SSC and SAM), continued

- Why the mixed license?
  - Want to encourage companies to use SSC and SAM as a foundation for growing their business in a fairly unrestricted way.
  - Want to encourage research institutions to share back any new innovations or make them publicly available so that the community as a whole benefits.
- Please see full license here:
  - https://github.com/NREL/SAM/blob/develop/LICENSE.md

Contributing to SAM

First Steps

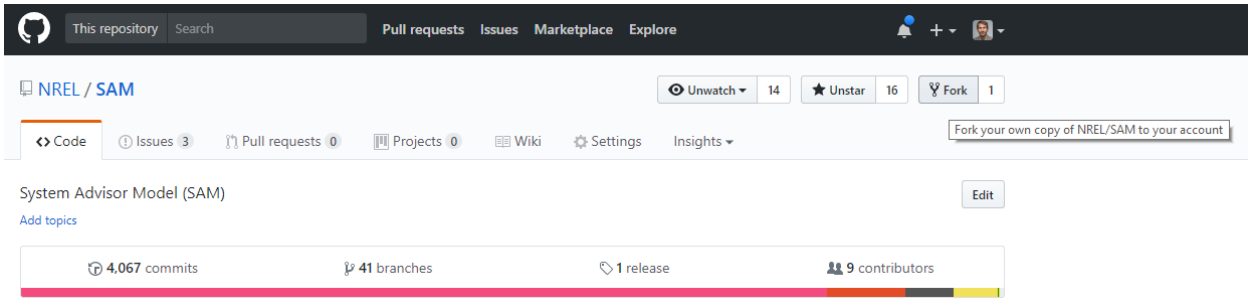
- Read contribution instructions:
  - https://github.com/NREL/SAM/blob/develop/CONTRIBUTING.md
- Send an email to nicholas.diorio@nrel.gov agreeing to the contribution policy.

Second Steps

- Scope your change and estimate how much time it will take. If the contributions are small (i.e, bug fixes), simply submit changes via pull request. If contributions are large, you will need to submit a description of the change for review. If the contribution fits within the project goals, we will work with you to create a plan to get the change incorporated.

Technical Contribution Process

1. Install your favorite Git client application



2. Create a fork for the repo of SAM you are contributing to (or to every SAM repo)

```
git clone https://github.com/NREL/lk.git
git clone https://github.com/NREL/wex.git
git clone https://github.com/NREL/ssc.git
git clone https://github.com/NREL/SAM.git
```

3. Clone your fork and the and build SAM according to instructions

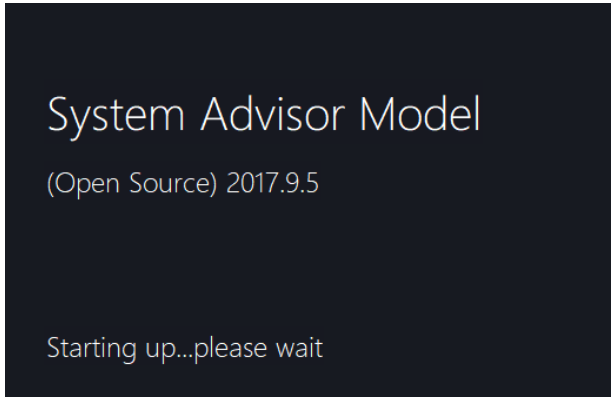
```
git checkout -b my_new_feature
```

4. Create a branch on the fork

Technical Contribution Process (2)

```
void battery_t::run(size_t idx, double I)
{
    // Compute temperature at end of timestep
    runThermalModel(I);
    runCapacityModel(I);
    runVoltageModel();
    runLifetimeModel(idx);
    runLossesModel(idx);
}
```

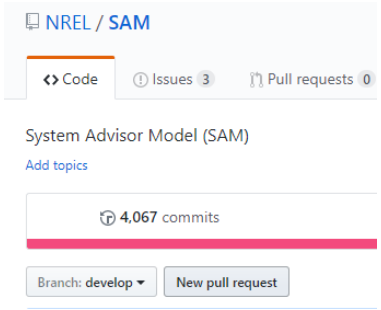
5. Make your code modifications



6. Build and test SAM. Fix compiler warnings, run simulations to test.

```
git commit -m "My new feature added"
git push
```

7. Commit and push changes to your branch



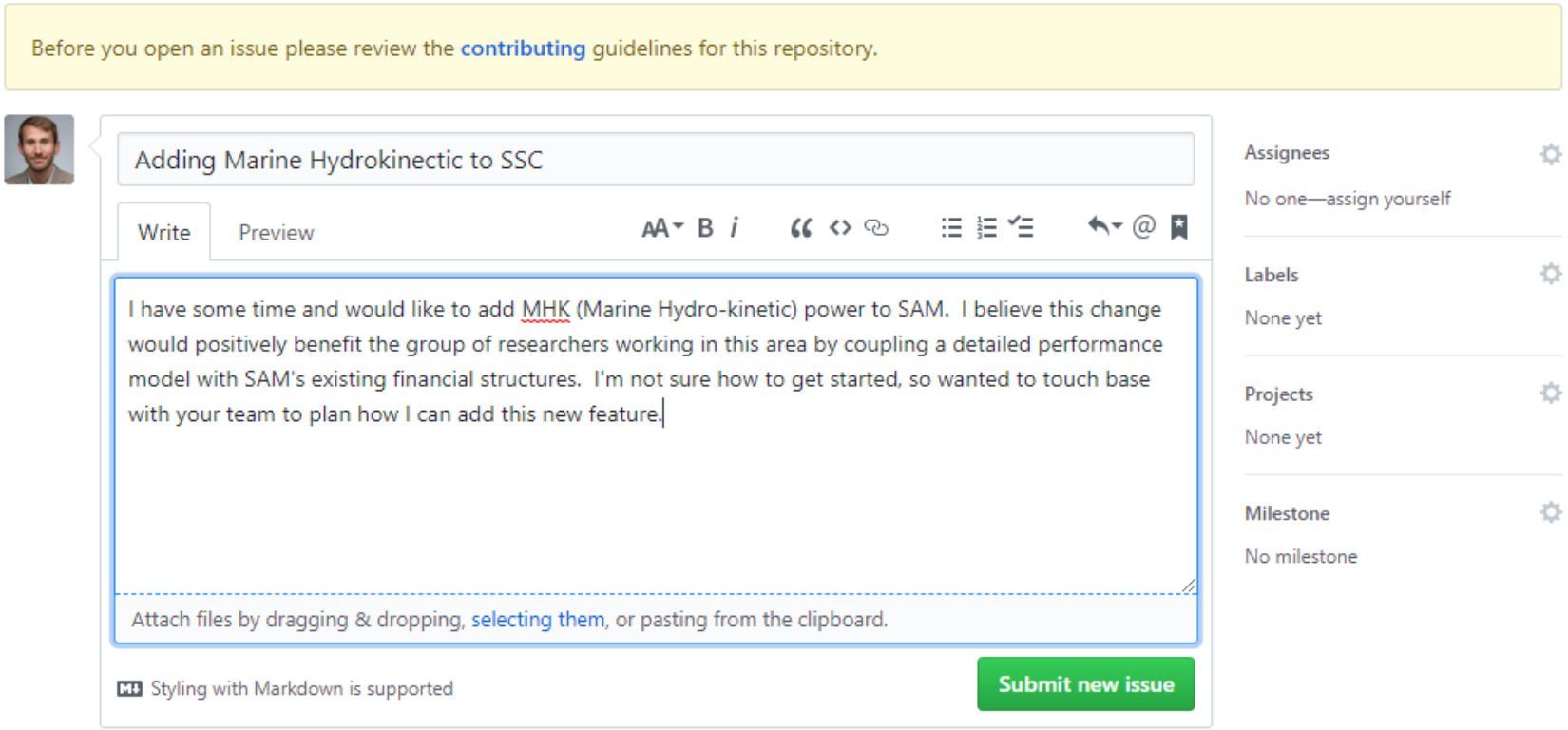
8. Start a pull request on GitHub, we will review, comment and merge in official version

Code quality

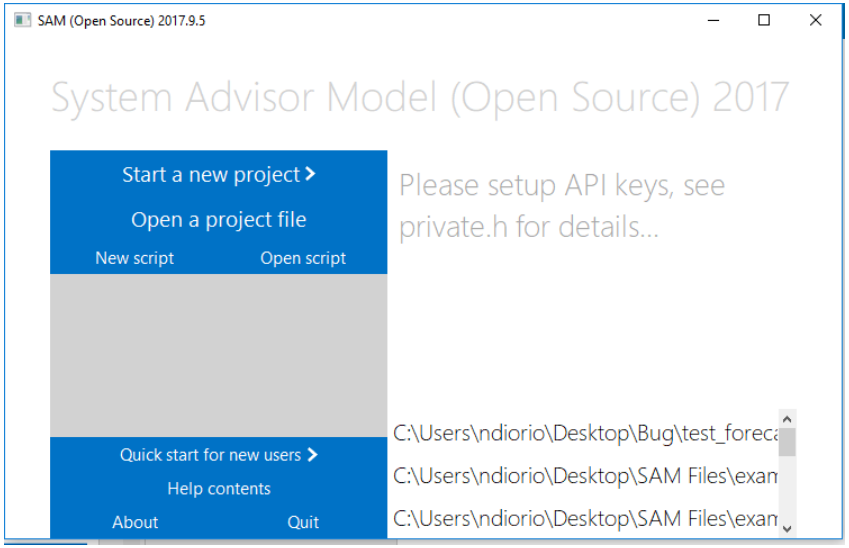
- **Testing**
  - We're in the process of getting GoogleTest setup for every repo (currently on SSC only).
  - We'd like substantial new contributions to be included with tests.
  - Please fix any compiler warnings that you introduce. SSC still has many warnings that need addressed.
- **Code Conventions**
  - SAM uses a mix of styles and conventions. We'd eventually like to standardize on one convention.
- **Documentation**
  - For substantial changes, please ensure you comment your code and provide documentation about what it does

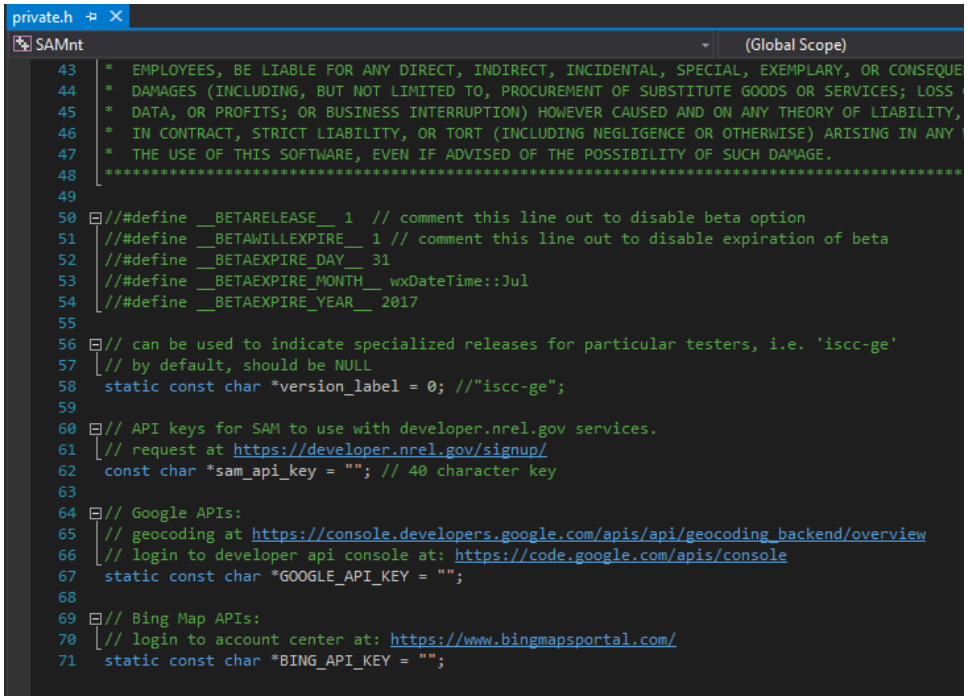
Issues tracking

If you discover a bug in the code, want to add a new feature, or have a question, use GitHub issues to tell us



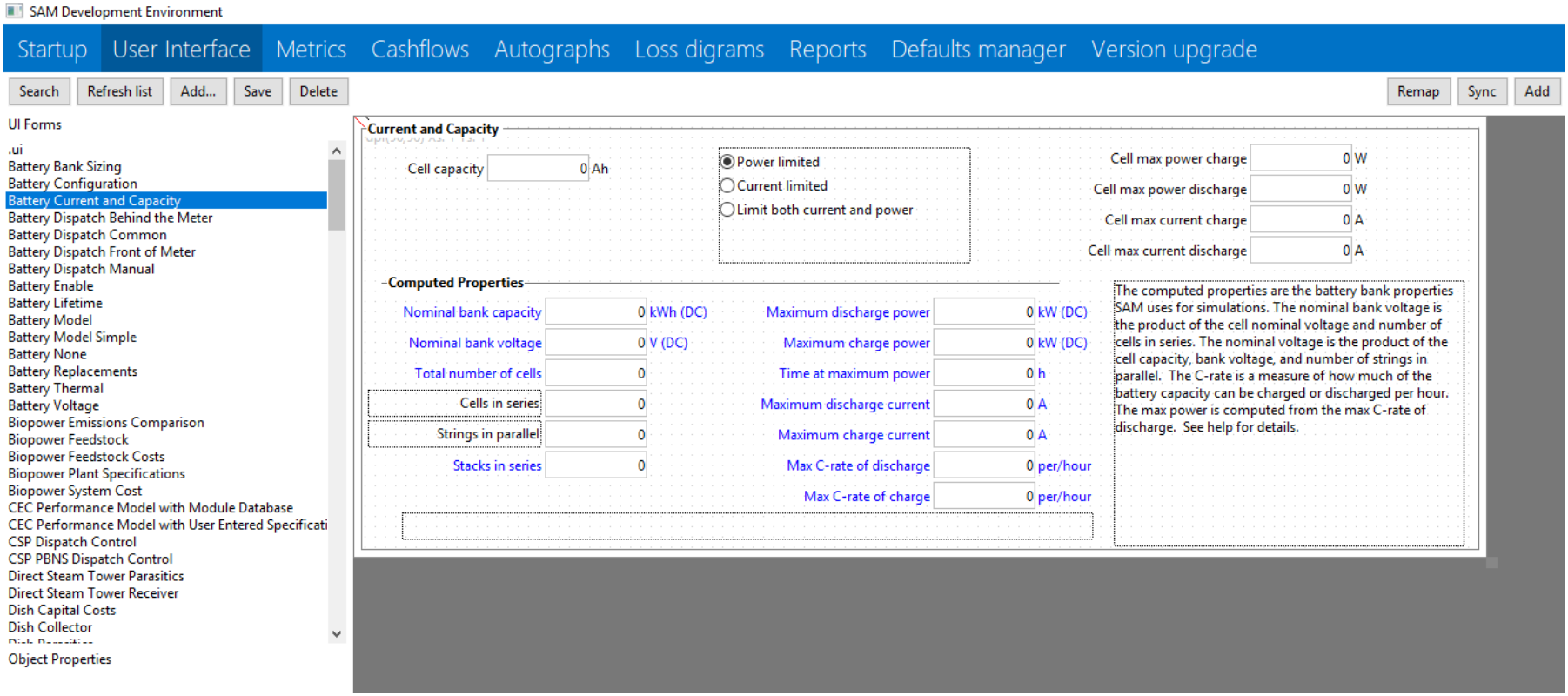
Setting up API Keys





- When you build SAM open-source, you'll need to get your own API keys setup
- Open the "private.h" file in the SAM project
- Go to the websites listed, and get the API keys. Paste them into the file between the empty quotes.
- Note, don't check in your API keys into the public repo!

Demo on editing user interface



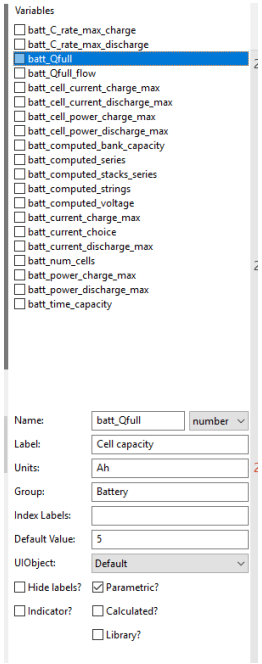
- To edit user interface, open the open source SAM executable.
- Press Shift + F7
- Make changes, save.
- Click “Startup” tab, and click restart (first make sure there are no open cases).
- You should see changes take affect

How do you use public tools?

- The Department of Energy is interested in learning how you use public tools and data (due date October 6th):
  - https://energy.gov/eere/sunshot/request-information-solar-energy-technology-analysis-data-needs
- Some examples of public tools and data:
  - SAM (NREL)
  - PVWatts (NREL)
  - NSRDB (NREL)
  - PVLIB (Sandia)

Questions?

Variables in the SAM UI and SSC




```
// Voltage discharge curve
{ SSC_INPUT,        SSC_NUMBER,      "batt_voltage_choice",          "Battery voltage input option",
{ SSC_INPUT,        SSC_NUMBER,      "batt_Vfull",                   "Fully charged cell voltage",
{ SSC_INPUT,        SSC_NUMBER,      "batt_Vexp",                    "Cell voltage at end of exponential zone",
{ SSC_INPUT,        SSC_NUMBER,      "batt_Vnom",                    "Cell voltage at end of nominal zone",
{ SSC_INPUT,        SSC_NUMBER,      "batt_Vnom_default",            "Default nominal cell voltage",
{ SSC_INPUT,        SSC_NUMBER,      "batt_Qfull",                   "Fully charged cell capacity",
{ SSC_INPUT,        SSC_NUMBER,      "batt_Qfull_flow",              "Fully charged flow battery capacity",
{ SSC_INPUT,        SSC_NUMBER,      "batt_Qexp",                    "Cell capacity at end of exponential zone",
{ SSC_INPUT,        SSC_NUMBER,      "batt_Qnom",                    "Cell capacity at end of nominal zone",
{ SSC_INPUT,        SSC_NUMBER,      "batt_C_rate",                  "Rate at which voltage vs. capacity curve input",
{ SSC_INPUT,        SSC_NUMBER,      "batt_resistance",              "Internal resistance",
{ SSC_INPUT,        SSC_MATRIX,      "batt_voltage_matrix",          "Battery voltage vs. depth-of-discharge",

// lead-acid inputs
```

SSC compute module variable table

SAM UI variables

- SAM user interface variables are read in by SSC compute modules.
- SSC compute modules are simply structures that encapsulate a specific model (i.e, PV, battery, utility rates)
- To export the current SAM case to be used in the SDKtool, or another language wrapper, hit “Shift+F5”. All of the SAM variables with their current values will be exported to a file in the language format you specify. Note, only variables defined in ssc.dll will be exported (need to build SSC if you add new variables).

Equations, and Callbacks in UI

Callbacks | Find... | Check code | Goto... | Help

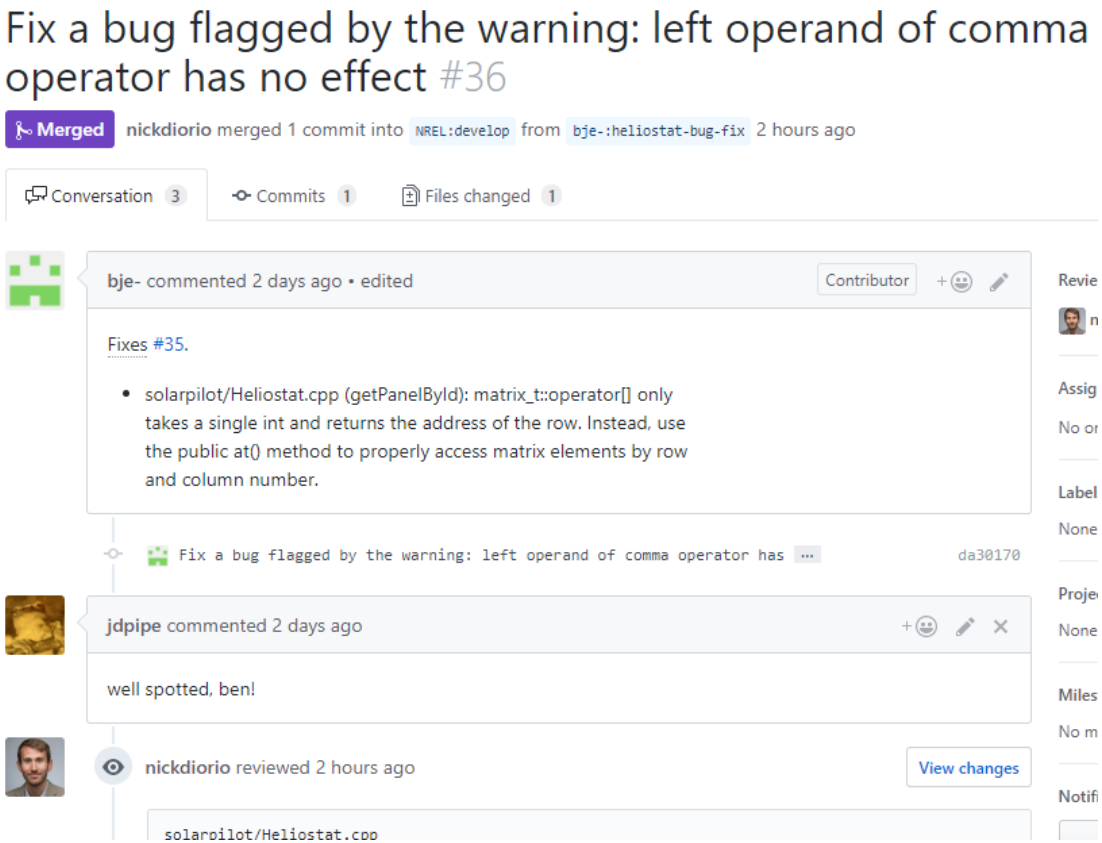
```
on_load{'Battery Current and Capacity'} = define()
{
    on_change{'batt_Qfull'};
    on_change{'batt_current_choice'};
    set_current_and_capacity();
};
on_change{'batt_Qfull'} = define()
{
    //update_voltage_plot(); // this makes a plot come up as a new window
    battsize_warning_check();
};
on_change{'batt_current_choice'} = define()
{
    set_current_and_capacity();
};

function battsize_warning_check()
{
    message = '';

    // if computed bank voltage is more than 1.5x desired, display
    if ((${batt_computed_bank_capacity} > 1.25*${batt_bank_size} ||
         ${batt_computed_bank_capacity} < 0.75*${batt_bank_size} ) &&
         ${batt_size_choice} == 0)
    {
```

Equations | Find... | Scan | Help

```
equations{ '$MIMO$ batt_bank_computed' } = define() {

    choice = ${batt_size_choice};
    isflow = ${batt_chem} == 2 || ${batt_chem} == 3;

    // if user input AC quantities, convert to DC
    batt_bank_power = ${batt_bank_power};
    batt_bank_capacity = ${batt_bank_size};

    if (${batt_bank_size_dc_ac} == 1)
        batt_bank_capacity /= (${batt_dc_ac_efficiency}*0.01);
    if (${batt_bank_power_dc_ac} == 1)
        batt_bank_power /= (${batt_dc_ac_efficiency}*0.01);

    // conventional battery
    if (!isflow)
    {
        batt_C_rate_max_discharge = 0;
        batt_C_rate_max_charge = 0;
        string_current = 0;
        bank_desired_voltage = 0;
        num_strings = 0;
        num_series = 0;
        num_parallel = 0;
```

- Each UI page has equations and callbacks.
- Callbacks are simply code that respond to user-interface events (loading the page, changing a variable input, etc.)
- Equations define “Calculated” variables and are updated anytime any of the inputs to the equation change.
- To access a UI variable you simply write **${variable_name}**
- MIMO equations are “multiple input, multiple output”, defining many equations at once

More on pull requests



- Offer a powerful way to review code changes
- Can provide comments inline
- Can approve your request, or ask for changes