



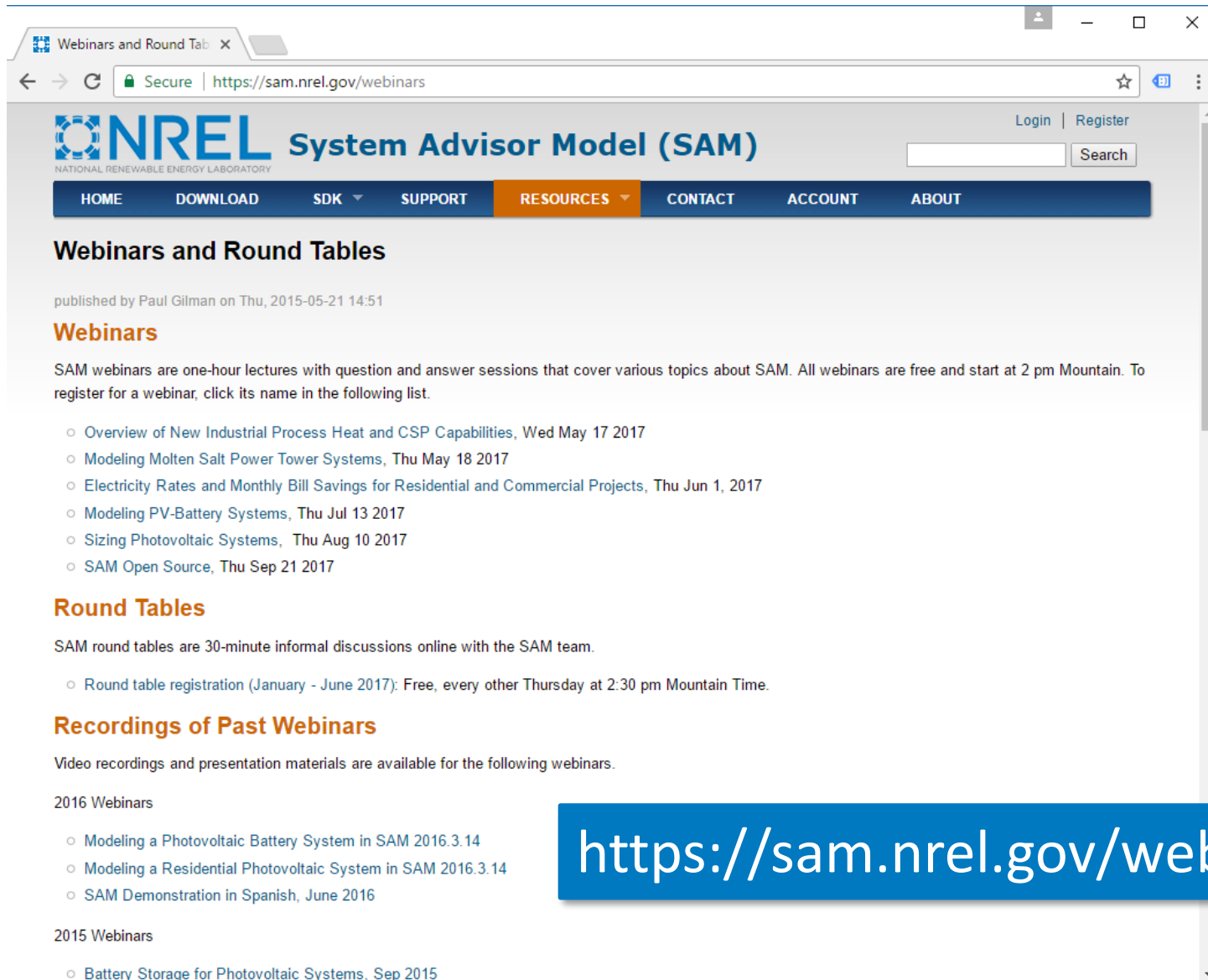
## **SAM Webinars 2017: SAM Open Source**

Nicholas DiOrio

September 21, 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



The screenshot shows a web browser window with the URL <https://sam.nrel.gov/webinars>. The page header includes the NREL logo and the text "System Advisor Model (SAM)". A navigation bar contains links for HOME, DOWNLOAD, SDK, SUPPORT, RESOURCES, CONTACT, ACCOUNT, and ABOUT. The main content area is titled "Webinars and Round Tables" and includes a sub-header "Webinars". It lists several webinars with their topics and dates, such as "Overview of New Industrial Process Heat and CSP Capabilities, Wed May 17 2017". Below this, there is a section for "Round Tables" and "Recordings of Past Webinars". A large blue box with the URL <https://sam.nrel.gov/webinars> is overlaid on the right side of the page.

Webinars and Round Tables

published by Paul Gilman on Thu, 2015-05-21 14:51

### Webinars

SAM webinars are one-hour lectures with question and answer sessions that cover various topics about SAM. All webinars are free and start at 2 pm Mountain. To register for a webinar, click its name in the following list.

- Overview of New Industrial Process Heat and CSP Capabilities, Wed May 17 2017
- Modeling Molten Salt Power Tower Systems, Thu May 18 2017
- Electricity Rates and Monthly Bill Savings for Residential and Commercial Projects, Thu Jun 1, 2017
- Modeling PV-Battery Systems, Thu Jul 13 2017
- Sizing Photovoltaic Systems, Thu Aug 10 2017
- SAM Open Source, Thu Sep 21 2017

### Round Tables

SAM round tables are 30-minute informal discussions online with the SAM team.

- Round table registration (January - June 2017): Free, every other Thursday at 2:30 pm Mountain Time.

### Recordings of Past Webinars

Video recordings and presentation materials are available for the following webinars.

#### 2016 Webinars

- Modeling a Photovoltaic Battery System in SAM 2016.3.14
- Modeling a Residential Photovoltaic System in SAM 2016.3.14
- SAM Demonstration in Spanish, June 2016

#### 2015 Webinars

- Battery Storage for Photovoltaic Systems, Sep 2015

<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**

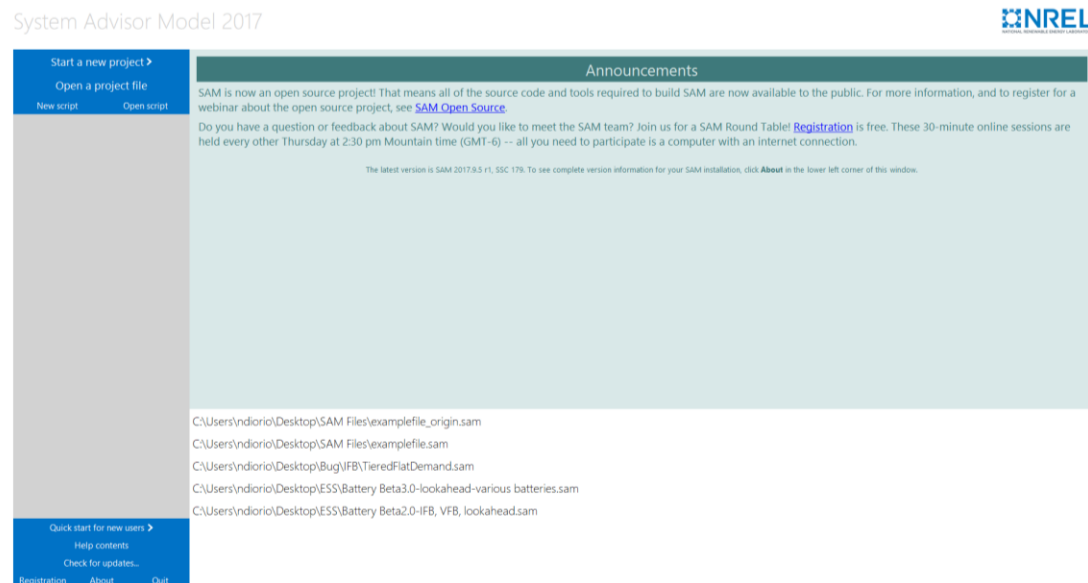
- Change the way a model works for research purposes
- Change electricity rate models to be specific to your country

## **Collaboration**

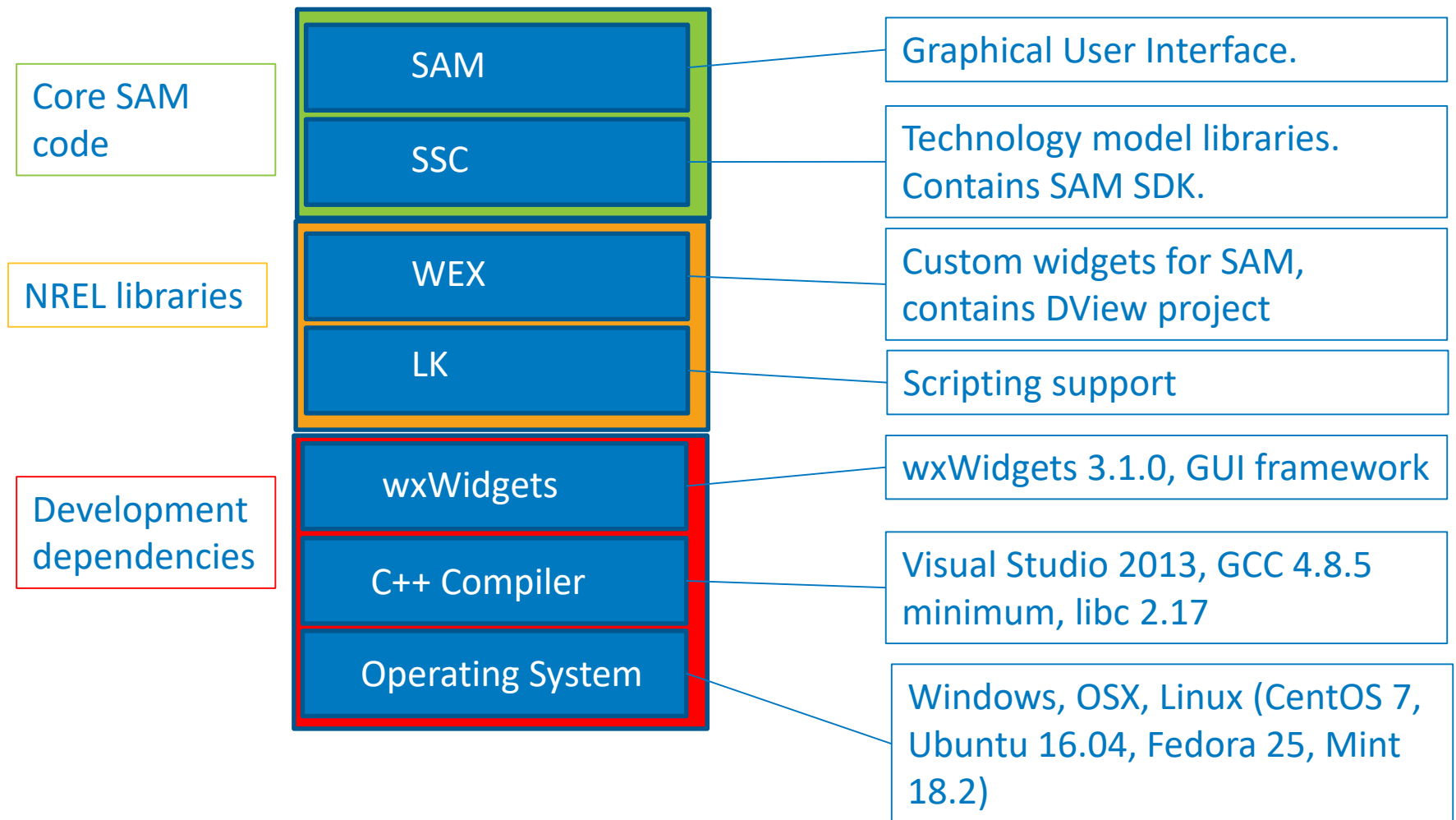
- Add new technology models
- Add a new battery dispatch model

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 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.

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

# Code Locations

<b>wxWidgets</b>	<a href="https://www.wxwidgets.org/downloads/">https://www.wxwidgets.org/downloads/</a>
<b>LK</b>	<a href="https://github.com/NREL/lk">https://github.com/NREL/lk</a>
<b>WEX</b>	<a href="https://github.com/NREL/wex">https://github.com/NREL/wex</a>
<b>SSC</b>	<a href="https://github.com/NREL/ssc">https://github.com/NREL/ssc</a>
<b>SAM</b>	<a href="https://github.com/NREL/SAM">https://github.com/NREL/SAM</a>

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>

## First Steps

- Read contribution instructions:
  - <https://github.com/NREL/SAM/blob/develop/CONTRIBUTING.md>
- Send an email to [nicholas.diorio@nrel.gov](mailto: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

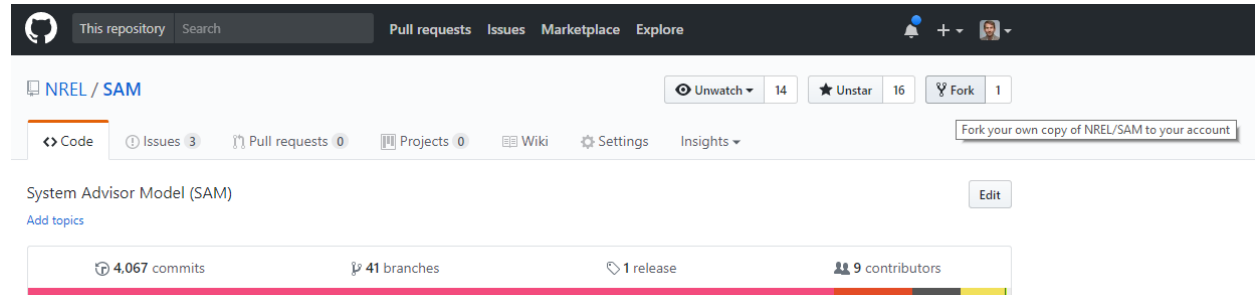


Git Bash

1. Install your favorite  
Git client application

```
git clone https://github.com/NREL/1kl.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



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

```
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

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

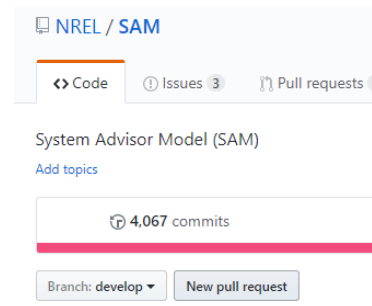
7. Commit and push changes to your branch

## System Advisor Model

(Open Source) 2017.9.5

Starting up...please wait

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



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



- **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

Before you open an issue please review the [contributing](#) guidelines for this repository.



Adding Marine Hydrokinetic to SSC

Write

Preview

AA ▾ B i “ < > ↺ ☰ ☷ ☹ ↶ @ 📎

I have some time and would like to add MHK (Marine Hydro-kinetic) power to SAM. I believe this change would positively benefit the group of researchers working in this area by coupling a detailed performance model with SAM's existing financial structures. I'm not sure how to get started, so wanted to touch base with your team to plan how I can add this new feature.

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

 Styling with Markdown is supported

Submit new issue

Assignees



No one—assign yourself

Labels



None yet

Projects



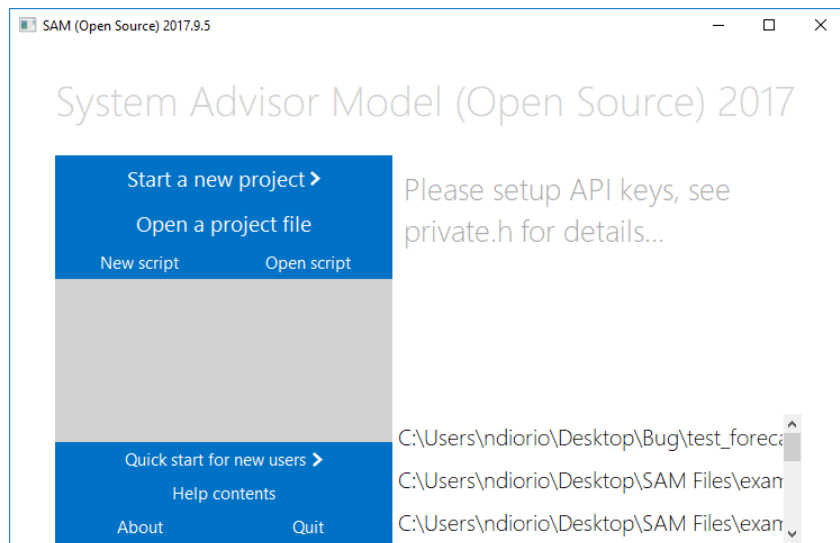
None yet

Milestone



No milestone

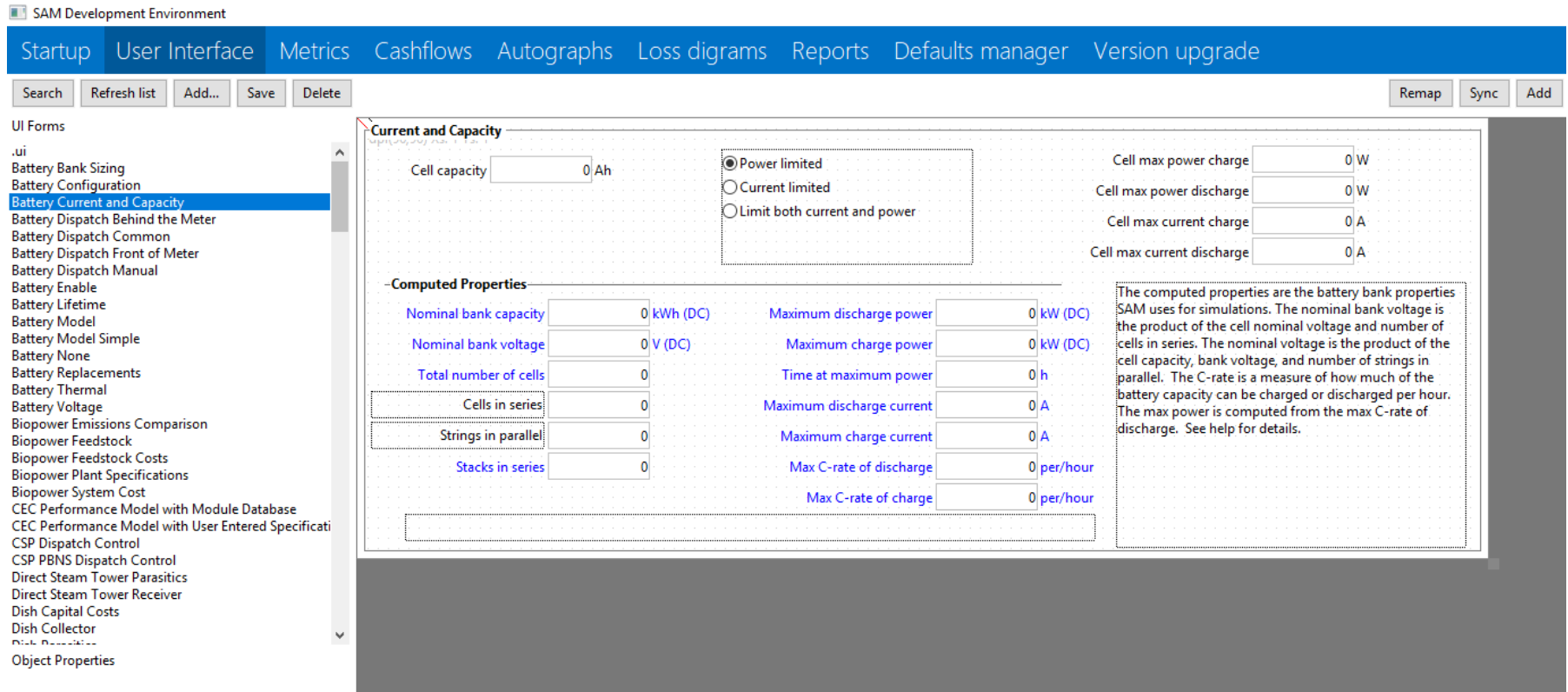
# Setting up API Keys



```
private.h  X
SAMnt (Global Scope)
43 * EMPLOYEES, BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENT
44 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
45 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
46 * IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
47 * THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
48 *****
49
50 // #define __BETARELEASE__ 1 // comment this line out to disable beta option
51 // #define __BETAWILLEXPRESS__ 1 // comment this line out to disable expiration of beta
52 // #define __BETAEXPRESS_DAY__ 31
53 // #define __BETAEXPRESS_MONTH__ wxDateTime::Jul
54 // #define __BETAEXPRESS_YEAR__ 2017
55
56 // can be used to indicate specialized releases for particular testers, i.e. 'iscc-ge'
57 // by default, should be NULL
58 static const char *version_label = 0; // "iscc-ge";
59
60 // API keys for SAM to use with developer.nrel.gov services.
61 // request at https://developer.nrel.gov/signup/
62 const char *sam_api_key = ""; // 40 character key
63
64 // Google APIs:
65 // geocoding at https://console.developers.google.com/apis/api/geocoding_backend/overview
66 // login to developer api console at: https://code.google.com/apis/console
67 static const char *GOOGLE_API_KEY = "";
68
69 // Bing Map APIs:
70 // login to account center at: https://www.bingmapsportal.com/
71 static const char *BING_API_KEY = "";
```

- 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 6<sup>th</sup>):
  - <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

The screenshot shows the SAM UI 'Variables' panel. A list of variables is on the left, with 'batt\_Qfull' selected. On the right, the configuration for 'batt\_Qfull' is shown:

- Name: batt\_Qfull (number)
- Label: Cell capacity
- Units: Ah
- Group: Battery
- Index Labels: (empty)
- Default Value: 5
- UI Object: Default
- Hide labels? ☐ Parametric? ☒
- Indicator? ☐ Calculated? ☐
- Library? ☐

```
// 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",

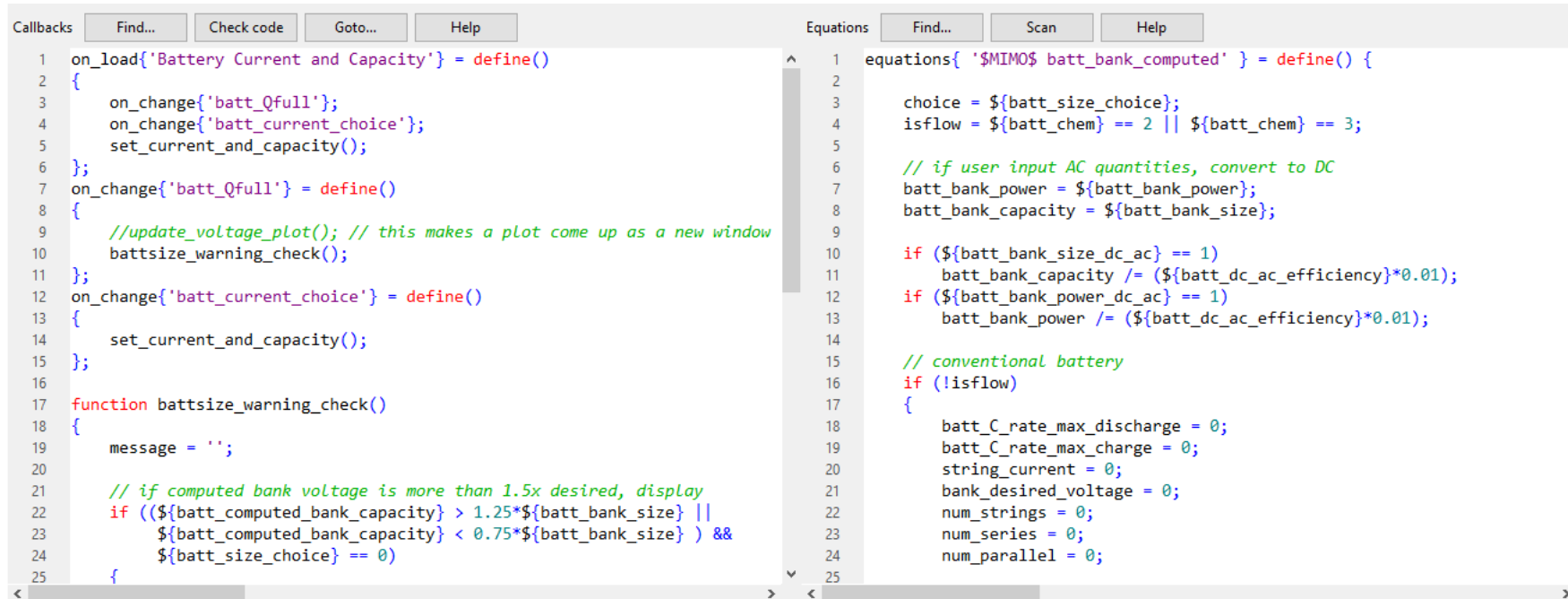
// lead-acid inputs
```

## 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
1 on_load{'Battery Current and Capacity'} = define()
2 {
3   on_change{'batt_Qfull'};
4   on_change{'batt_current_choice'};
5   set_current_and_capacity();
6 };
7 on_change{'batt_Qfull'} = define()
8 {
9   //update_voltage_plot(); // this makes a plot come up as a new window
10  battsize_warning_check();
11 };
12 on_change{'batt_current_choice'} = define()
13 {
14   set_current_and_capacity();
15 };
16
17 function battsize_warning_check()
18 {
19   message = '';
20
21   // if computed bank voltage is more than 1.5x desired, display
22   if (($batt_computed_bank_capacity > 1.25*$batt_bank_size ||
23     $batt_computed_bank_capacity < 0.75*$batt_bank_size) &&
24     $batt_size_choice == 0)
25   {
26
27   }
28 }

Equations
Find... Scan Help
1 equations{'$MIMO$ batt_bank_computed'} = define() {
2
3   choice = ${batt_size_choice};
4   isflow = ${batt_chem} == 2 || ${batt_chem} == 3;
5
6   // if user input AC quantities, convert to DC
7   batt_bank_power = ${batt_bank_power};
8   batt_bank_capacity = ${batt_bank_size};
9
10  if (${batt_bank_size_dc_ac} == 1)
11    batt_bank_capacity /= (${batt_dc_ac_efficiency}*0.01);
12  if (${batt_bank_power_dc_ac} == 1)
13    batt_bank_power /= (${batt_dc_ac_efficiency}*0.01);
14
15  // conventional battery
16  if (!isflow)
17  {
18    batt_C_rate_max_discharge = 0;
19    batt_C_rate_max_charge = 0;
20    string_current = 0;
21    bank_desired_voltage = 0;
22    num_strings = 0;
23    num_series = 0;
24    num_parallel = 0;
25  }
```

- 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

## Fix a bug flagged by the warning: left operand of comma operator has no effect #36

**Merged** nickdiorio merged 1 commit into NREL:develop from bje-:heliostat-bug-fix 2 hours ago

Conversation 3 Commits 1 Files changed 1

The screenshot displays a GitHub pull request interface. At the top, a green box indicates the pull request is 'Merged'. Below this, a summary shows 'nickdiorio merged 1 commit into NREL:develop from bje-:heliostat-bug-fix 2 hours ago'. A navigation bar shows 'Conversation 3', 'Commits 1', and 'Files changed 1'. The main content area shows a comment from 'bje-' (green robot icon) stating 'Fixes #35.' and listing a change in 'solarpilot/Heliostat.cpp'. Below the comment is a commit by 'da30170' with the message 'Fix a bug flagged by the warning: left operand of comma operator has no effect'. A comment from 'jdpipe' (yellow robot icon) says 'well spotted, ben!'. At the bottom, a review by 'nickdiorio' (man icon) is shown with a 'View changes' button. On the right side, a sidebar contains a 'Review' section with a dropdown menu showing options: 'Assign', 'None', 'Label', 'None', 'Project', 'None', 'Miles', 'No m', and 'Notifi'.

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