Introduction and Overview of SAM Open Source and SDK

Nicholas DiOrio

August 28, 2019
## Schedule for Wednesday, August 28, 2019

All times are in U.S. Mountain time.

<table>
<thead>
<tr>
<th>Time</th>
<th>Title</th>
<th>Speaker</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:00 - 10:30 am</td>
<td>Introduction and Overview of SAM Open Source and SDK</td>
<td>Nicholas DiOrio</td>
<td>National Renewable Energy Laboratory (NREL)</td>
</tr>
<tr>
<td>10:30 - 11:00</td>
<td>PySAM: An Improved Python Wrapper</td>
<td>Darice Guittet</td>
<td>NREL</td>
</tr>
<tr>
<td>11:00 - 11:20</td>
<td>Integrating SAM with SunDAT</td>
<td>Jaya Rajesekaran</td>
<td>FTC Solar</td>
</tr>
<tr>
<td>11:20 - 11:40</td>
<td>Integrating SAM with a Web Portal for PV Consumers in Saudi Arabia</td>
<td>Russ Jones</td>
<td>KA Care</td>
</tr>
<tr>
<td>11:40 - 12:00 pm</td>
<td>Integrating SAM for Rural Microgrid Load Modelling</td>
<td>Brian Somers</td>
<td>Standard Microgrid</td>
</tr>
<tr>
<td>12:00 - 12:20</td>
<td>Hydrogen Production From Variable Renewable Energies</td>
<td>Yassine Ennassiri</td>
<td>Moroccan Agency for Sustainable Energy (MASEN)</td>
</tr>
<tr>
<td>12:20 - 12:30</td>
<td>Wrap Up</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Motivation

Open Source
• Inspect the underlying implementation of performance or financial models
• Add custom functionality or additional features for your projects
• Contribute and become part of the community.

Software Development Kit
• Wrap SAM in your project and access models of interest via public interfaces.
• Abstract out the things you don’t want to reinvent so you can focus on what’s important to you

```cpp
class cm_pvwattsV5_base : public compute_module
{
  protected:
    double dc_nameplate, dc_ac_ratio, ac_nameplate, inv_eff_percent;
    double loss_percent, tilt, azimuth, gamma;
    bool use_ar_glass;
    int module_type;
    int track_mode;
    double inocpt;
    int shade_mode_1x;
    int array_type;
    double gcr;
```
Outline

• SAM Open Source
  o Repositories
  o Contributing
  o Licenses

• Software Development Kit (SDK)
  o Languages supported
  o Getting started
SAM Events 2019

• Introduction to PySAM, August 1, 2019
• Modeling Wind Systems in SAM, August 22, 2019
• SAM Virtual Conference, August 28, 2019
• Modeling Fuel Cells in SAM, September 19, 2019

Learn about upcoming events here:
https://sam.nrel.gov/events.html
System Advisor Model (SAM)

SAM is free software for modeling the performance and economics of renewable energy projects.

http://sam.nrel.gov
github.com/NREL/SAM

- Developed by NREL with funding from DOE
- Windows, OSX, and Linux
- One or two new versions per year
- Software Development Kit (SDK)
- Support

NREL actively maintains and releases new versions of SAM compiled from the open source repositories.
SAM Open Source Code Architecture

Core SAM code (C++11)

NREL libraries (C++11)

Development dependencies

Graphical User Interface.

Technology model libraries. Contains SAM SDK.

Custom widgets for SAM, contains DView project

Scripting support

wxWidgets 3.1.0, GUI framework

Visual Studio 2013, GCC 4.8.5 minimum, libc 2.17

Windows, OSX, Linux (CentOS 7, Ubuntu 16.04, Fedora 25, Mint 18.2)
## Code Locations

<table>
<thead>
<tr>
<th>Code Location</th>
<th>Repository URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>wxWidgets</td>
<td><a href="https://www.wxwidgets.org/downloads/">https://www.wxwidgets.org/downloads/</a></td>
</tr>
<tr>
<td>LK</td>
<td><a href="https://github.com/NREL/lk">https://github.com/NREL/lk</a></td>
</tr>
<tr>
<td>WEX</td>
<td><a href="https://github.com/NREL/wex">https://github.com/NREL/wex</a></td>
</tr>
<tr>
<td>SSC</td>
<td><a href="https://github.com/NREL/ssc">https://github.com/NREL/ssc</a></td>
</tr>
<tr>
<td>SAM</td>
<td><a href="https://github.com/NREL/SAM">https://github.com/NREL/SAM</a></td>
</tr>
</tbody>
</table>

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

---

**LK, WEX, SSC, and SAM are now all licensed under BSD-3-clause**
Accepting contributions

Contribution Policy
Janine edited this page 5 days ago • 7 revisions

SAM Contribution Policy

The SAM team welcomes your contribution to the project!

If you contribute code, you agree that your contribution may be incorporated into SAM and made available under the SAM license in compliance with standard GitHub terms of service.

In the effort to make SAM easy to contribute to, the contribution process is composed of three simple steps:

1. Fork the relevant SAM repositories.
2. Make your modifications. For large new features, we suggest that you email us before you start working so that we coordinate your contribution (see more info for large contributions below!).
3. Create a pull request.

Additional contribution help

Some additional details on what makes it more likely that your code will be readily accepted into the main development version of SAM.

- For large contributions
- Code quality and testing
- Documentation
- For further assistance

Best part of open-source is that anyone can contribute back to the code. Goal is to make this as easy as possible for everyone involved!
Code quality

• **Testing**
  – We use GoogleTest
  – We’d like substantial new contributions to be included with tests.
  – Please fix any compiler warnings that you introduce.

• **Code Conventions**
  – Please try and broadly adhere to the same style of code in the area you will be working

• **Documentation**
  – For substantial changes, please ensure you comment your code and provide documentation about what it does
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

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.
Building SAM open source

Quick Steps for Building SAM

For detailed build instructions see the wiki, with specific instructions for:

- Windows
- Mac
- Linux

- Process outlined on GitHub for all platforms
- In the process of supporting CMake for all projects
- In the process of making LK, WEX, SSC submodules of the SAM repository for a more unified experience
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!
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 effect
The SAM Simulation Core (SSC) software development kit (SDK) is a collection of developer tools and documentation for creating renewable energy system models using the SSC library. The SDK allows you to create your own applications using the SSC library.
Supported language wrappers

- LK for SDK tool
- C
- C#
- Python 2 and 3
- Java
- MATLAB
- PHP 5 and 7
- SAM inputs to JSON
- Android Studio
- XCode Swift for iOS
Generating inputs to SDK

Workflow
- Create a case in SAM desktop application that reflects the default model you want
- Click on the case drop down and select “Generate code”.
- Select the language wrapper of choice
- SAM will output the library (.dll/.so), as well as the relevant language code and inputs which you can then wrap in your code.
**SDK overview**

**sscapi.h** - Exposes available calls into SSC

**Populate data**

ssc_data_t : the underlying type holding data inputs and outputs

ssc_data_create() : return a ssc_data_t

ssc_data_set_number( ssc_data_t, const_char * varname, double value)

... other setters

**Define and run model**

ssc_module_t : an underlying SSC model (like PVWatts)

ssc_module_create(“module”) : return a ssc_module_t

ssc_module_exec(ssc_module_t module, ssc_data_t data): run the model with the given data

**Extract data**

ssc_data_get_number( ssc_data_t, const_char * varname, double * value)

.. Other getters

*Each language wrapper is a set of functions that provide access to the underlying C-api using methods supported in the language*
```php
$dat = sscphp_data_create();
sschp_data_set_string( $dat, "solar_resource_file", "C:/SAM/2018.11.11/solar_resource/blythe_ca_33.617773_-114.588261_psmv3_60_tmy.csv" );
sschp_data_set_number( $dat, "system_capacity", 249 );
sschp_data_set_number( $dat, "module_type", 0 );
sschp_data_set_number( $dat, "dc_ac_ratio", 1.25 );
sschp_data_set_number( $dat, "inv_eff", 96 );
sschp_data_set_number( $dat, "losses", 14.075660705566406 );
sschp_data_set_number( $dat, "array_type", 0 );
sschp_data_set_number( $dat, "tilt", 15 );
sschp_data_set_number( $dat, "azimuth", 180 );
sschp_data_set_number( $dat, "gcr", 0.40000000596046448 );
sschp_data_set_number( $dat, "adjust:constant", 0 );
$mod = sscphp_module_create("pvwatts5");
api_call_to_pvwatts -> return 8760 generation profile
populate $dat, "gen" with pvwatts->ac*1000
if ( !sschp_module_exec( $mod, $dat ) )
{
    $index = 0;
    while ( $err = sscphp_module_log( $mod, $index++ ) )
    {
        echo "$err" . PHP_EOL;
    }
    exit;
}
sschp_module_free( $mod );
```
Questions?