eSiMon User Manual

Prepared for the
Office of Science
U.S. Department of Energy

A. Khan, S. Klasky, P. Mouallem, N. Podhorszki,
C. Silva, R. Tchoua, M. Vouk

January 31st, 2011

Prepared by

OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 379831-6070
managed by
UT-BATTELLE. LLC
for the
U.S. DEPARETMENT OF ENERGY
under contract DE-AC05-00OR22725

Table of Contents

Table of Figures

Acknowledgements

The eSiMon dashboard is a joint product of the National Center of Computational Sciences (NCCS) at Oak Ridge National Laboratory (ORNL), the North Carolina State University (NCSU) and the Scientific Computing and Image Institute (SCI) at the University of Utah. This work is being led by Scott Klasky, Mladen Vouk and Claudio Silva. The main contributors are Roselyne Tchoua, Norbert Podhorszki, Ayla Khan, Pierre Mouallem, Bradley Grimm and Mei Nagappan.

eSiMon was designed, implemented and tested with greatly appreciated help from theoretical scientists. Special thanks to Seung-Hoe Ku and Prof. C.S. Chang at New York University and Julian Cummings at the California Institute of Technology.

This project is sponsored by ORNL, the Scientific Data Management (SDM) Center, and the Center for Plasma Edge Simulation (CPES).

eSiMon contributors:

NCSU: Guruprasad Kora, Pierre Mouallem, Mei Nagappan, Nagiza Samatova, Mladen Vouk

ORNL: Scott Klasky, Norbert Podhorszki, Roselyne Tchoua

University of Utah: Ayla Khan, Bradley Grimm, Emanuele Santos, Claudio Silva

I. Introduction

The live version of the eSiMon dashboard is physically located at Oak Ridge National Laboratory (ORNL) at https://esimmon.ccs.ornl.gov. This is a secure site not accessible without a National Center for Computational Sciences account. However eSiMon is available for download at http://www.nccs.ornl.gov/user-support/esimmon.

eSiMon stands for electronic Simulation Monitoring. We use eSiMon for documentation purposes however for practical reasons esimmon is used throughout the functions, files and directories.

The purpose of the eSiMon dashboard is to be an at a glance view of the status and health of a simulation. For example, they can view the status of the supercomputers and interact with the simulation datasets. The key is the ease of use. eSiMon attempts to reduce the information technology overhead and allow scientists to individually or collectively focus on, and exchange of ideas on their science.

The code for eSiMon was developed using Flex, framework to create Flash Applications. The code is written in ActionScript and MXML. Many of the physical phenomena in simulations happen over a certain period of time and it is useful to visualize and animate the data across time stamps (time steps). Flash creates dynamic pages with local interaction and asynchronous server communication that allows flexible data manipulation on the back end. The Flash Player is a virtual machine that runs the same way in different browsers. In that sense using Flash represents provides consistency and familiarity and there is no need to develop and to maintain browser specific code. Other Flash features include videos, vector graphics and 3D graphics.

In the current version of the eSiMon dashboard there are two major groups of functionalities: Machine Monitoring and Simulation Monitoring. On the live eSiMon, the machine monitoring page presents scientists with the status and queues of the participating Department of Energy (DOE) supercomputers at the National Center for Computational Sciences (NCCS) and the National Energy Research Scientific Computing (NERSC) Center. Users can monitor the state of their runs: active, eligible or queued. They can customize this view depending on their systems of interest and other preferences such as font and browser size. Collaborators can see each other's runs (or *shots*) and annotate their shots for themselves and others. When monitoring running jobs, users have access to images, which they can chose to view as they are being updated. They can also go back and forth between past time steps.

Viewing an old job offers many more options. In this tutorial we will present the most basic of these options. Users can visualize variables by simply dragging and dropping them from a Tree View of simulation variables. They do need to know about any specific graphic tools. As they drag and drop variable names, eSiMon links the variable string with the corresponding Flash movie (*flv*) file and returns a movie. The movies are xy-plots or 2D slices of 3D variables. Scientists can annotate movies or make electronics notes on simulations. The simulation page features a space for more extensive and general notes on the simulation. Other capabilities include visualization of the data provenance information (e.g., full path of the raw data), downloading of the processed and/or raw data. They can manipulate their shots as well as those of their collaborators. We will explore the common monitoring features of past

simulation shots in section2. Section 3 describes a set of PHP API to feed information to the dashboard. We start with the installation steps in Section 1.

II. Installation

A. Obtaining eSiMon

eSiMon can be found at http://users.nccs.gov/~rbarreto/esimmon
This website contains the information needed on how to download, setup, and use eSiMon.

B. Installation (UNIX)

To install eSiMon, execute the following steps:

1. Using a web browser, navigate to http://users.nccs.gov/~rbarreto/esimmon and click on "Download eSiMon". This will download the setup script (buildesimmon.sh)
2. Move the script to the user's home directory on the target machine. Alternatively you can directly download the setup script onto the target machine using the following command:
   wget http://users.nccs.gov/~rbarreto/esimmon/download/buildesimmon.sh
3. Change the permissions buildesimmon.sh by running the following command
   chmod +x buildesimmon.sh
4. Get usage information for the script
   sudo ./buildesimmon.sh -h
5. Run the script as root (optionally with flags)
   sudo ./buildesimmon.sh
6. Follow the onscreen instructions and wait until the script finished execution. The script will download and install all the packages required for the application. The script will also check the compiler versions to insure the correct building of all the packages. The installation takes place in 3 phases:
   a. Check the version of c, c++, ruby and python compilers available in the target machine.
   b. Build the basic software that eSiMon and the associated tools depend on, namely zlib, libpng, mxml, MPI.
   c. Build the tools used by eSiMon, namely NetCDF, ADIOS, Mplayer, FFmpeg, FLVTool2, Xmgrace, libcurl.
   d. The optional SRM-Lite package is installed and the java run time environment required for it is verified.
   e. The eSiMon source code
7. Once it is done, navigate using a web browser to **http://<youripaddress>/esimmon** to view the eSiMon dashboard (make sure you replace <your IP address> with the actual IP address of the machine you just ran the script on). You may be asked to download the Flash Media Player for your browser upon entering the URL.

C. Dependencies

The eSiMon package includes the core of the code: the Graphical User Interface (Flash application) and the server code used to implement the basic eSiMon features. However without a monitoring API that

feeds information from a simulation to the eSiMon database, it remains only a front-end application. In other words, users would have to understand and implement methods to give it content. We plan to provide C API and enable users to automatically connect their running simulations to eSiMon. In this case, we depend on additional software to generate Flash (.flv) movies from images generated by the simulation. Moreover, our approach to eSiMon has been to provide a single access point to users' analysis tools. In future releases, we plan to provide and support hooks into external software for data management and analysis. This alpha version of eSiMon includes a hook into a Storage Resource Manager SRM-Lite.

1. Data Description Software

a. NetCDF:

NetCDF (network Common Data Form) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.

b. AdIOS:

The Adaptable IO System provides a simple, flexible way for scientists to describe the data in their code that may need to be written, read, or processed outside of the running simulation. By providing an external to the code XML file describing the various elements, their types, and how you wish to process them this run, the routines in the host code (either FORTRAN or C) can transparently change how they process the data.

2. Movie Generation Software

a. Mplayer/Mencoder:

Mplayer is a movie player for Linux and many other Unix distributions. Mencoder is a converter that supports many input video format.

b. FFmpeg:

FFmpeg is a complete, cross-platform solution to record, convert and strean audio and video. It is used to generate movies from eSiMon.

c. FLVTool2:

FLVTool2 is a manipulation tool for Macromedia Flash Video files (FLV). We use it to add cur points to the .flv files and keep track of simulation time steps.

Note: FLVTool2 has a Ruby dependency. The tagging script for the movies is written in Python.

3. Storage Resource Manager

SRM-Lite allows efficient data movement across wide area network. It is a simple command-line based tool with pluggable file transfer protocol supports developed at the Lawrence Berkeley National Laboratory (https://sdm.lbl.gov/twiki/bin/view/Software/SRMLite). It is Java-based. The PHP and C API explain how to register provenance information about images generated during the simulation on the eSiMon database. Provided that the path to SRM-Lite is set and the data lineage information is available in the eSiMon data store, users can conveniently download raw simulation data files from the monitoring simulation page.

Note: SRM-Lite has a Java dependency.

II. Basics

A. Usage

ESiMon is installed onto your system along with XAMPP. XAMPP is a free and open source cross-platform web server package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages. To get started you should always have the server up and running. This will happen automatically at installation. If you reboot the server, you will need to restart the server as root: **/opt/lampp/lampp start**

For 64-bit Red Hat Enterprise Linux (RHEL), CentOS or other Linux operating systems with Security-Enhanced Linux (SELinux) enabled, it may be necessary to run the command **setenforce 0** before starting the server.

For more information on running/stopping the server see http://www.apachefriends.org/en/xampp-linux.html.

B. Login

Browse to the eSiMon dashboard at http://youripaddress/esimmon or http://localhost/esimmon. If you are a new user, click on the "New User?" link above the Login Button to get to the registration page.

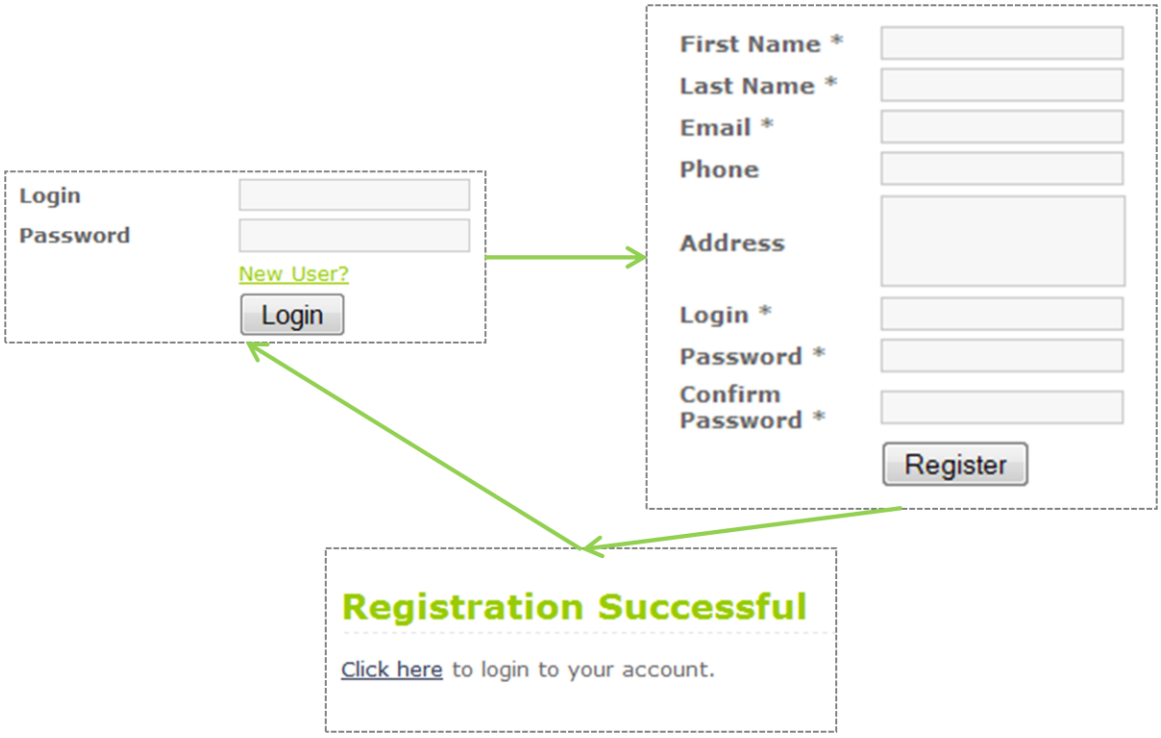

Figure 1: Logging in.

After logging in, you get the eSiMon dashboard main (home) page shown in Figure 2. The page is divided in two main sections; the top part is a series of tab showing your jobs while the bottom tabs show your collaborator's jobs.

As a new user, you will not have running, eligible or past jobs. Instead you will automatically get user ***mycol*** as a collaborator and thus be able to view one of this user's simulation past runs (*shots*). In addition you will learn to manipulate an "old" simulation shot. Double clicking on one of your running jobs or your collaborators' running job would take you to a similar interface simply with fewer features. In the Running Job Monitoring Page users can view images of variables updating at each time step of the simulation, while on the Past Jobs Simulation Page they manipulate movies of variables evolving through time. The first one is more passive and as previously mentioned includes less features, therefore we will focus on the latter one.

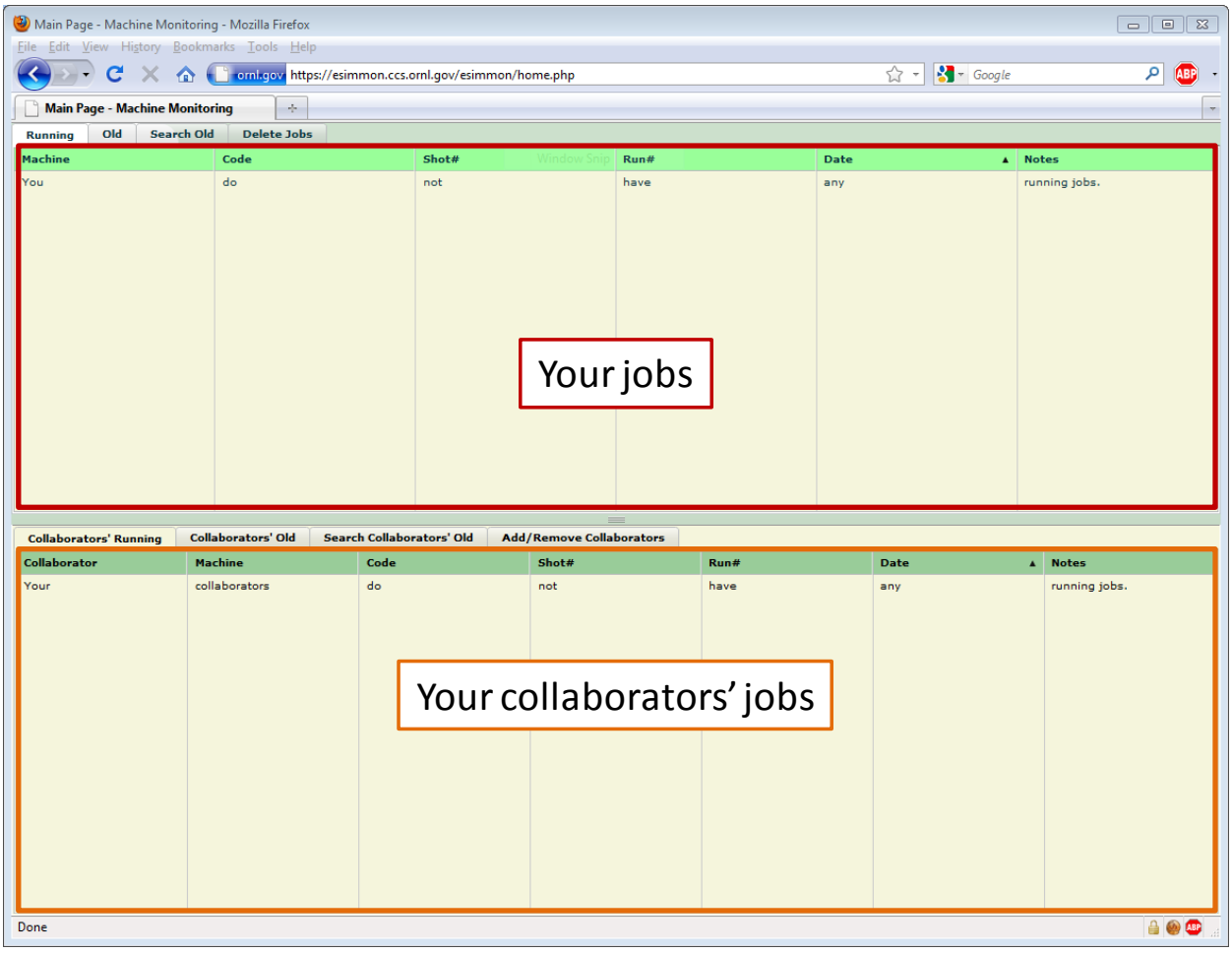

Figure 2: Home Page

C. Collaborators

1. Add/Remove Collaborators

eSiMon provides ways for you to allow other users to view your shots. On your collaborators' section (left-bottom section) of the home page:

a. Click on the "***Add/Remove Collaborators***" tab.
b. As an example to allow another registered eSiMon user **sklasky** to view your shots, you would find "sklasky" on the list of eSiMon users,
c. select the check box by the username
d. then click on the "**Save**" button

Figure 3 : Add/Remove Collaborators

You can give access to your shots to several users at the time. The current way to request access to other users' shots is to contact them and ask them to perform these same steps with your username. As previously mentioned, user ***mycol*** has already automatically been added to your list of collaborators.

2. Collaborators Old

Click on "**Collaborators' Old**" tab to view a grid list of all your collaborators' past shots. Each rows display information about the shots including: username, machine name, code name, shot name, run name or number, date and short notes entered by the owner of the shot. Users enter notes from their own list view of past shots in the upper right section of the home page by simply editing the corresponding grid cell. Having two, three or more collaborators can easily increase the number of shots on this list. Therefore the "Search Collaborators' Old" allows you to search through the same list of shots. You can search old shots by usernames, machine names, shot numbers and notes.

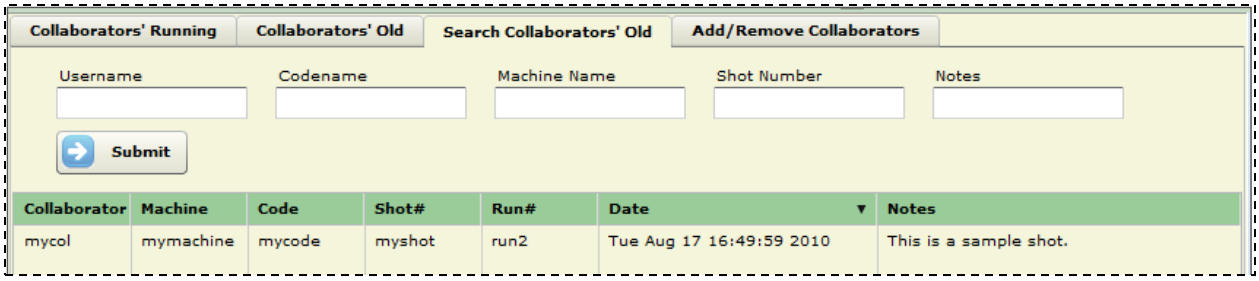

Figure 4: Collaborators' Old Jobs

D. Simulation Monitoring Page

To view your collaborators' shot **select** and **double click** on any shot from any shot listed under your shots or your collaborators (running, old, and search old tabs). A collaborator with a sample shot has been added to your username at first login. We will use this shot to illustrate the basic simulation monitoring features available for old shots. From the Collaborators' Old or the Search Collaborators' Old tabs double click on user mycol only shot (myshot). The data for this specific shot has been downloaded during the installation. There are three main components to this view (Figure 5): the Application Control Bar on top of the page, the Tree View of Variable on the left with option to search for a specific variable in the tree below it and the Main Canvas.

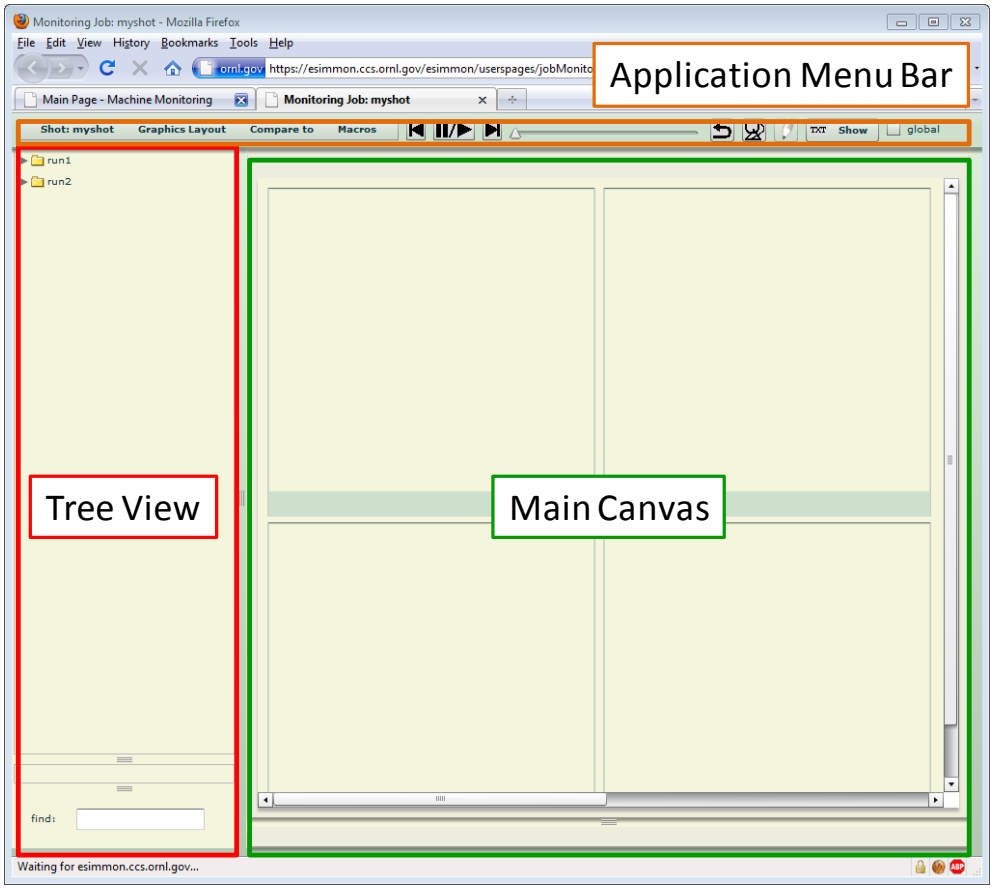

Figure 5: Simulation Monitoring View

1. Tree View of Variables

a. **Click** on the **arrow next to the folders** of the Tree View to **expand/close** them. Click on the **"run1"** folder to view the variables generated during the first run of this shot.

b. **Drag** (Click and Hold) variable "***cos***" from the Tree View and **Drop** (Release) it on the **first cell** (top left) in the Main Canvas

c. Repeat for variable "***sin***", this time drop it on the cell to the right of the first cell

d. On the text field below the tree, start typing "***tan***". You will reduce the tree to 2 variables that contain the string ***"tan"***

e. Drag and Drop "***run1/comp/tan***" in the cell below the first cell

f. **Click** on the ***Find*** text field below the tree view to return to the full Tree View

2. Movies

a. On the Application Control Bar locate the "***VCR***" buttons.

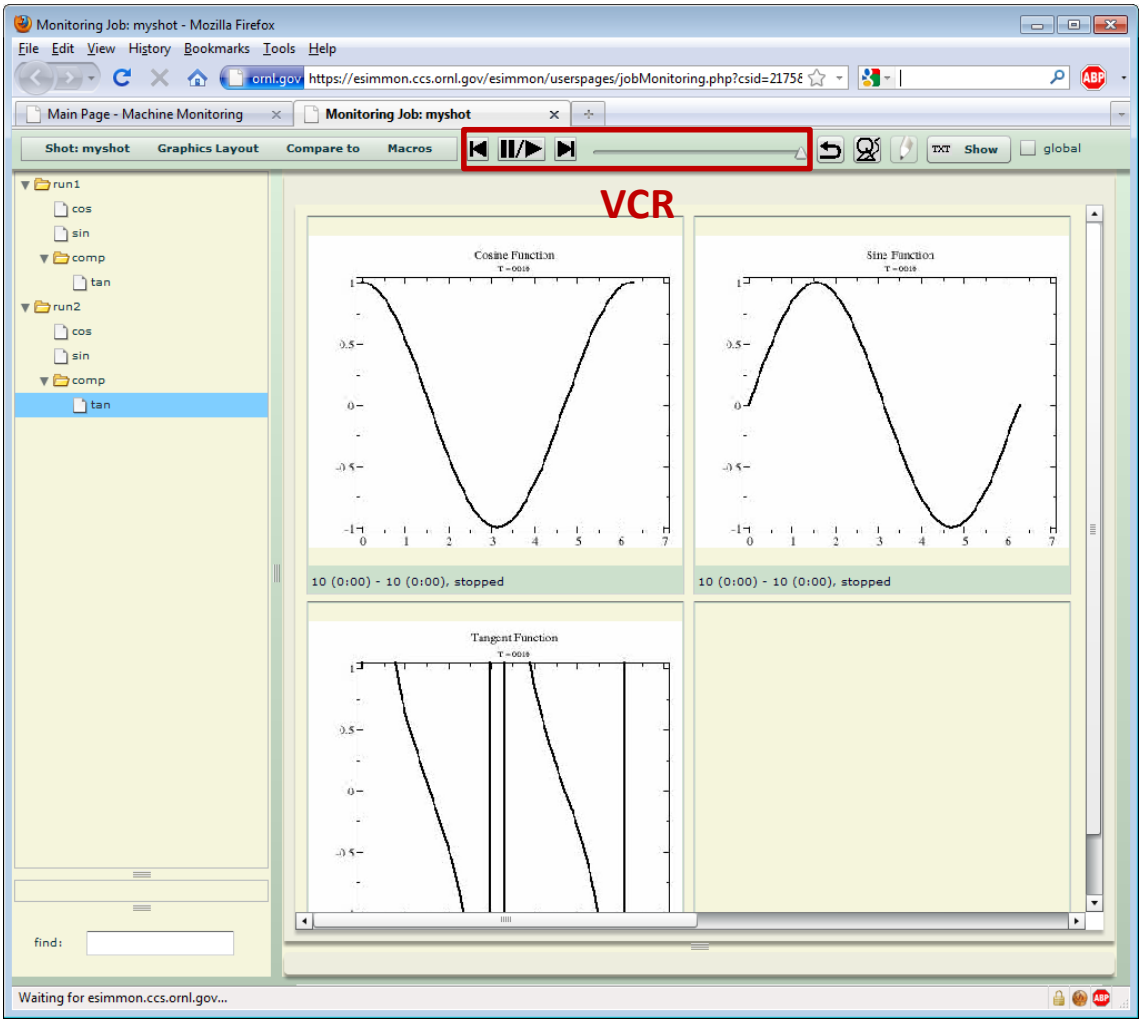

Figure 6: Playing Movies

Figure 7: VCR Buttons

b. **Click** on the "**Play/Pause**" button
c. **Move** the Time Slider Thumb to Time Step **5** for example. The tool tip on the slider indicates your position
d. **Step** forward, **step** backward using the corresponding buttons
e. **Toggle** the "Auto/Rewind" button and then click on the "**Play/Pause**" button again. Let it play until the last time step. When the movie is finished playing, it will rewind back to the first time step.
f. **Double Click** on the ***"sin"*** movie. A pop-up window opens, automatically playing the movie in its original size. You can also navigate through this movie using the slider.
g. When you are done with this movie, **close** the pop-up window using either button with an "X" icon on the Window (Top and bottom right corners)

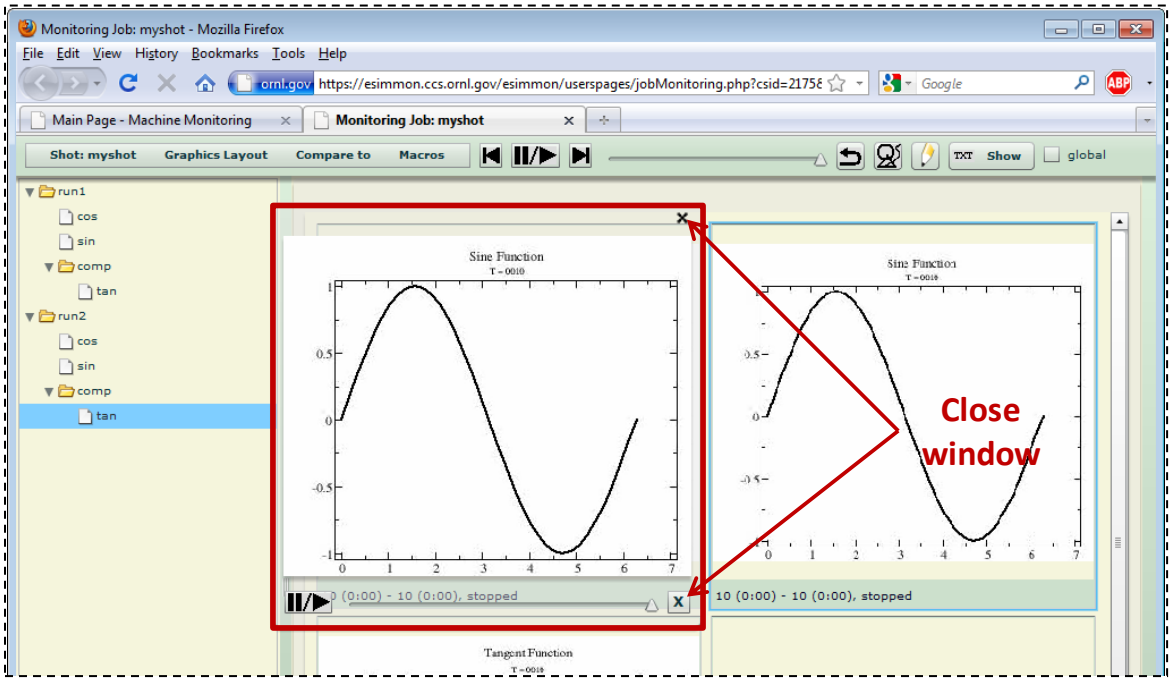

Figure 8 :Pop-Up Window

Note: The last button on the Application Menu Bar is a checkbox labeled "**global**". Some simulations/workflows generate variables without knowing the minima and maxima a priori. In these

cases the compilation of images will appear discontinuous as the axis will change at each time step. Therefore there will be a need for newly generated images and movies at the end of the simulation with a common global maximum and minimum for x and y axis. The same is true for color maps and 2D variables. Such movies should be placed under a "**global/**" directory under each variable directory. When this is done, users can check/uncheck the global checkmark to switch back and forth between local and global movies.

3. Layout

The next section explores the "Graphics Layout" menu item of the Application Menu Bar.

The Main Canvas is composed of five different spaces, four spaces with Video Cells for the presentation of movies and one space with two Text Areas for file viewing. By default, every user starts on *Space 1.*

a. Click on **Graphics Layout** ➔ **Space** ➔ **Space 2** to load a new space
b. Click on **Graphics Layout** ➔ **Windows** ➔ **4 windows** to change the number of video cells
c. Adjust the size of your windows by using **Graphics Layout** ➔ **Resize**. A *Resize* pop-up window appears with the current size of the cells
d. Approximate a new desired size for your four windows and type it in the ***width*** text field and click **OK**

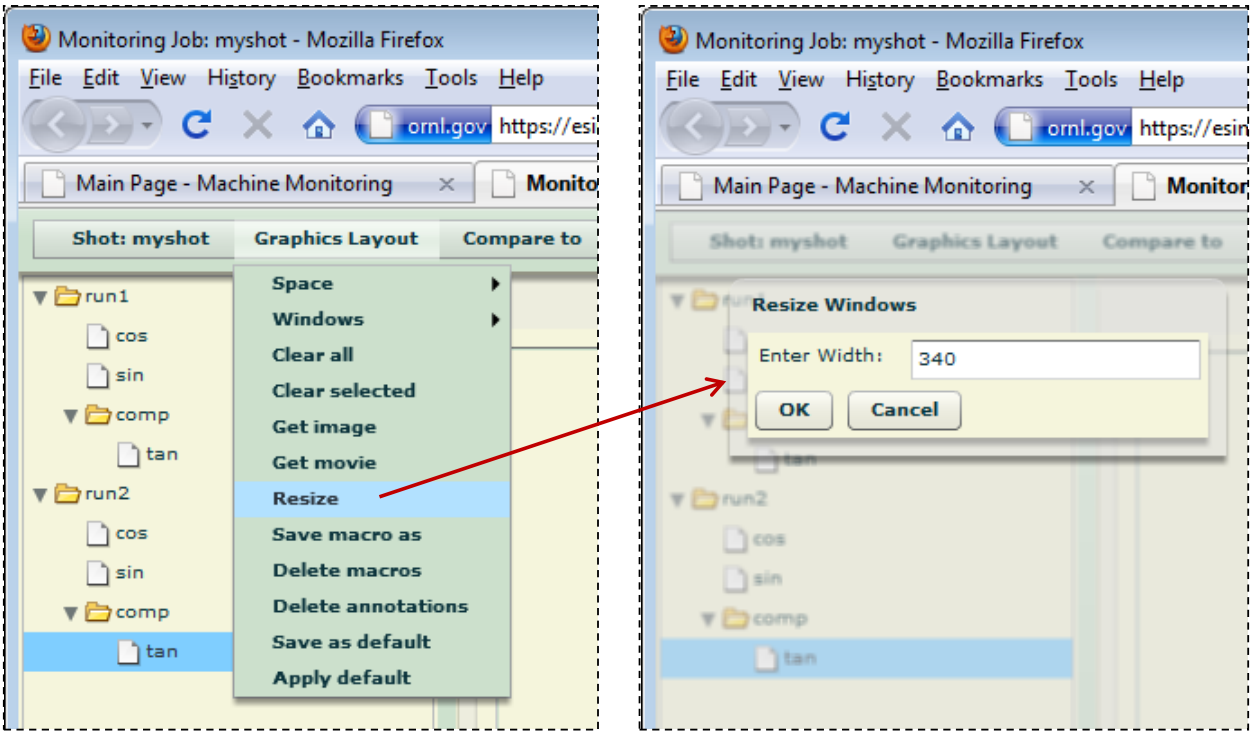

Figure 9: Resize Pop-Up Window

e. Repeat if necessary.

f. When you have set a layout which satisfies you, providing you often use the same browser and computer you may want to set it as your default layout for every future shot. To do so, select **Graphics Layout ➔ Save as default**

g. Optional: Load a few variables of your choice onto this space, play them, and repeat a few other steps from section C 2 for practice.

h. Select **Graphics Layout ➔ Space ➔ Space 3** and notice that eSiMon now loads your custom default layout instead of the default 9 windows layout.

4. Macros

In the case where you are used to look at a specific set of variables together, you can save this particular assortment of variable as a ***Macro***. This combination of variables will remain available on your settings when you open up a different simulation run of the same code naturally.

a. From the Application Control Bar **click** on the "***Macros***" menu item. Notice that you do not have any macros created yet on eSiMon

b. Return to space 1 and our initial selection of variables.

c. Select **Graphics Layout ➔ Save macro as**

d. A new pop-up window appear, type a name for your macro (without spaces), then click **OK**.

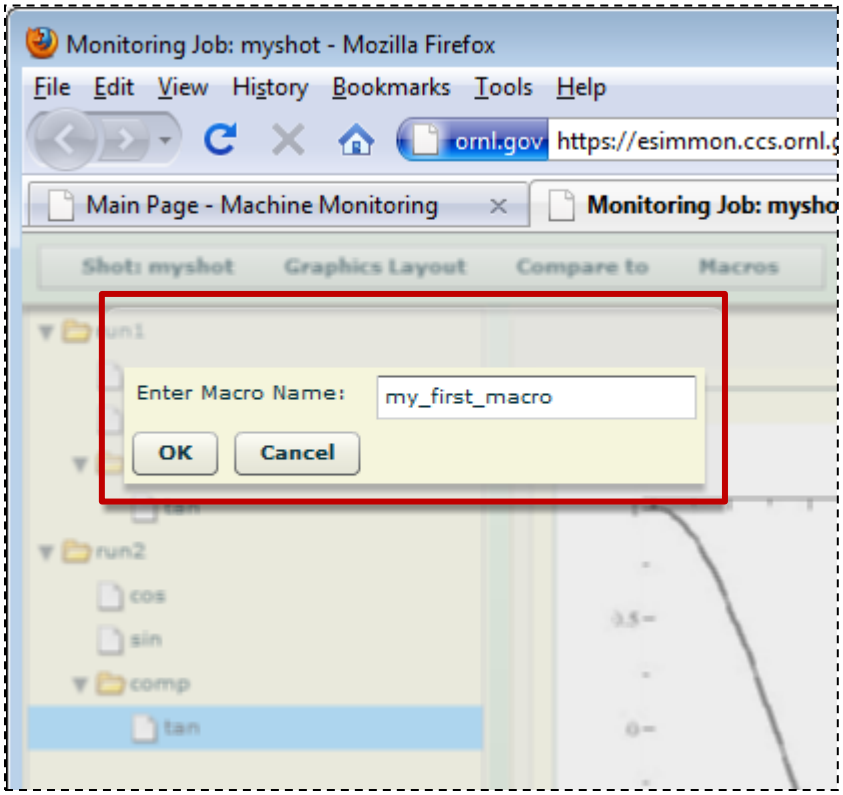

Figure 10: Saving a Macro

e. Now that you have created your first macro, select **Graphics Layout ➔ Space ➔ Space 4**

f. You would normally apply the same macro to another shot. For illustration purpose we will apply this macro to Space 4. Select **Macros** ➔ ***yournewmacroname*** and load your newly created macro to Space 4

5. Select and Clear Cell(s)

On the Main Canvas, some operations can target a specific cell after it is selected by a mouse click. A select cell is highlighted in blue.

a. **Select** the first cell (top left corner)
b. To clear only this movie: select **Graphics Layout** ➔ **Clear selected**

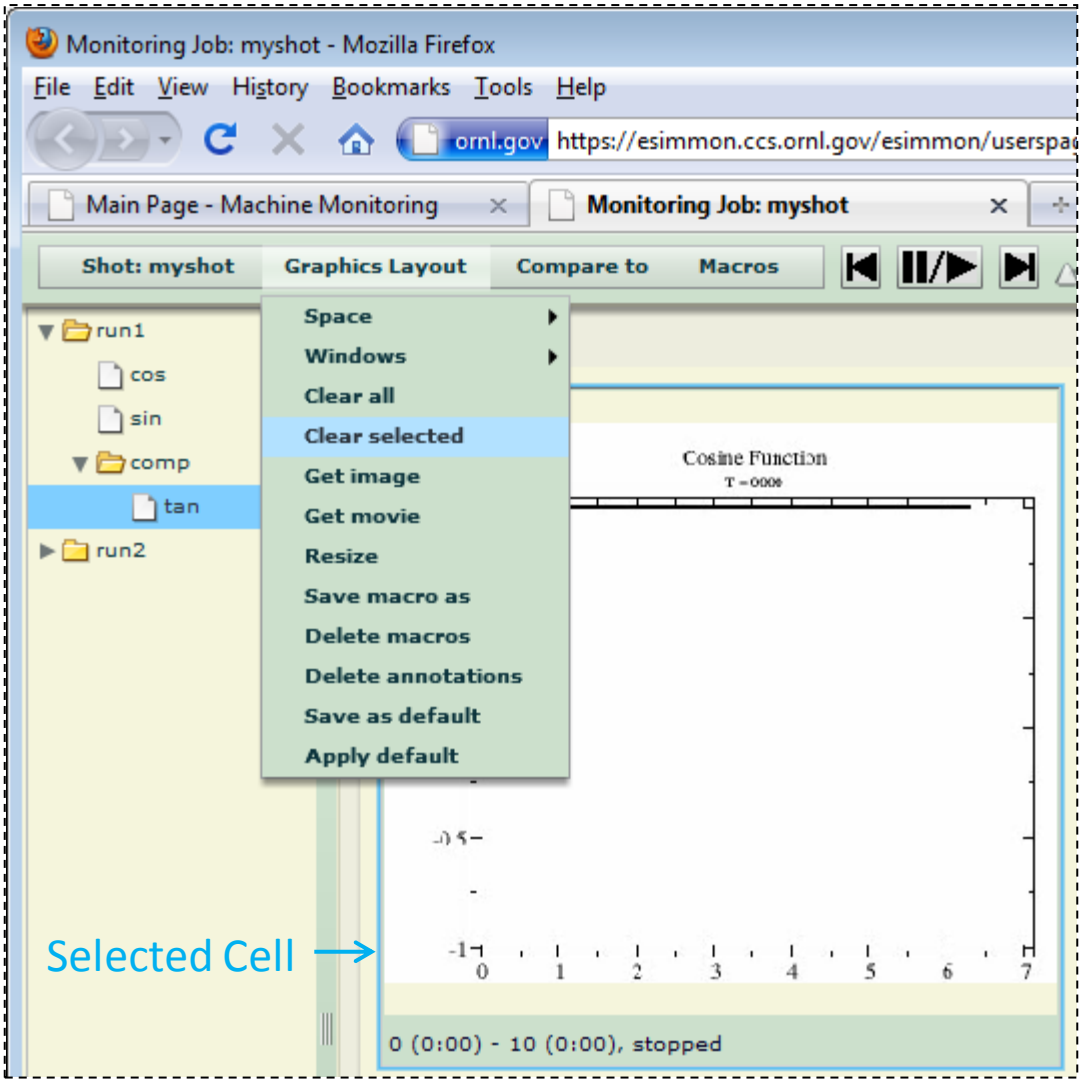

Figure 11: Main Menu Clear Movie

c. **Right-click** on the green section or the Video Cell Status Bar (displays frame number and status of video) of the second cell. **Select** **Clear Movie**
d. To clear all cells: select **Graphics Layout** ➔ **Clear all**

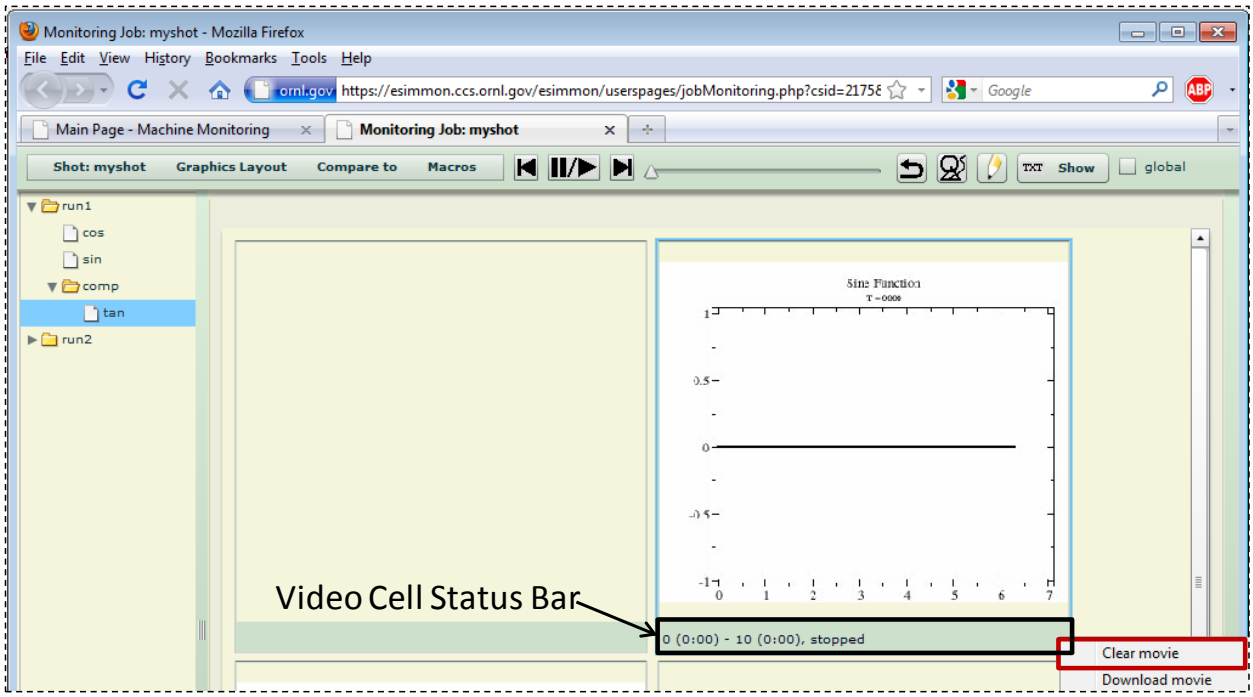

Figure 12: Right-Click Clear Movie

6. Download image and or movie

a. **Drag** variable "*cos*" back onto the first cell

b. **Select** it, move the slider to any time step of your choice

c. To **download** the image for this time step, select **Graphics Layout** ➔ **Get Image**

d. **Confirm** your selection by clicking **Yes**

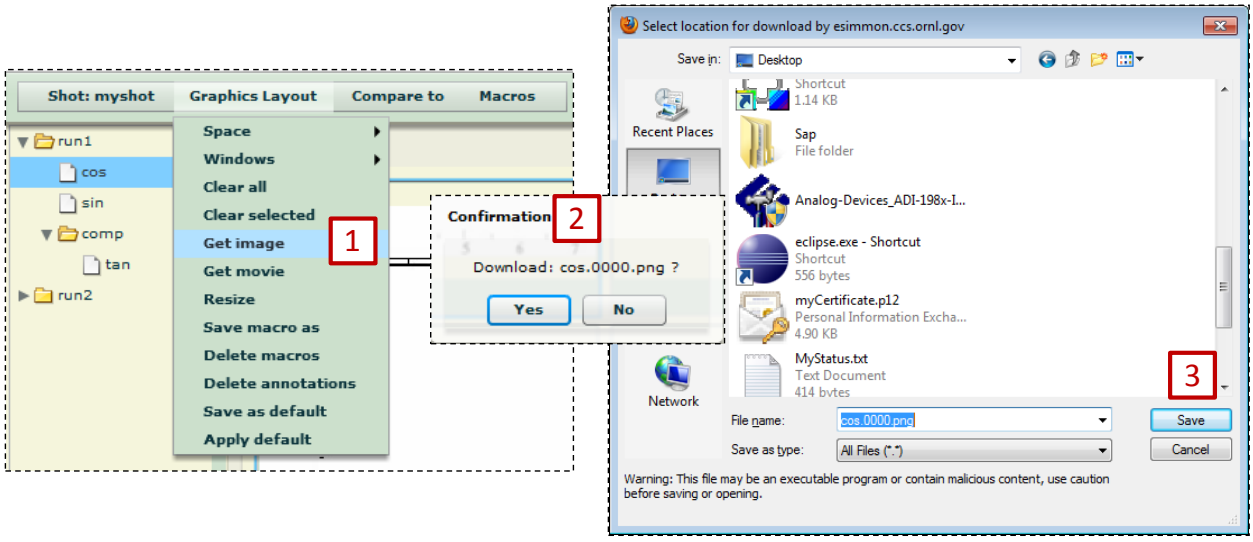

Figure 13: Main Menu Download Image

e. The same action can be performed by **right-clicking** on the Video Cell Status Bar and selecting **Download Image**

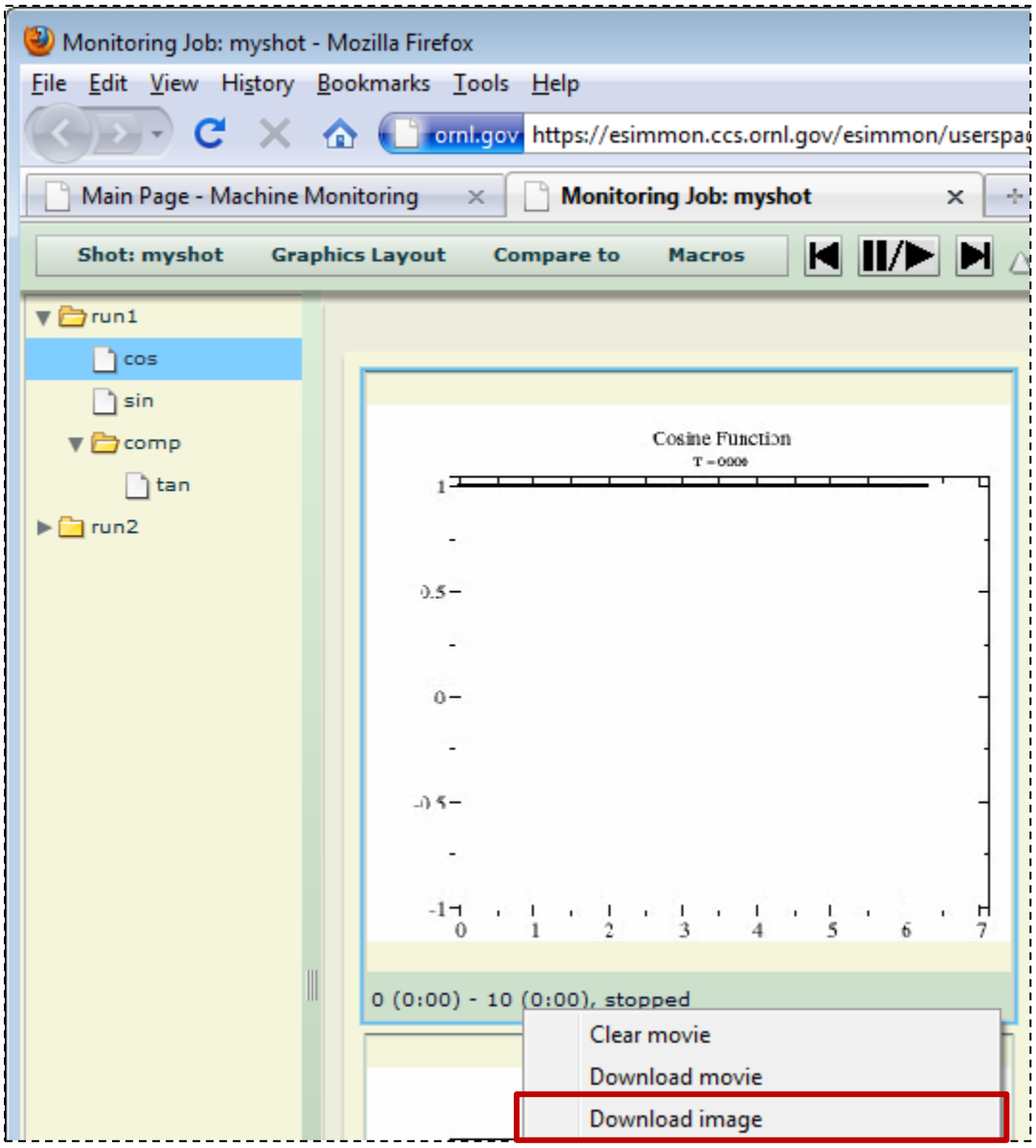

Figure 14: Right-Click Download image

f. **Repeat steps c and e**, this time selecting **Graphics Layout** ➔ **Get Movie** from the Application Control Bar and/or **Download movie** from the Video Cell Right-Click Menu.

Note: that the default format for the video is .avi. For loading into a presentation, simply rewrite the movie extension as wmv before saving the file. Then insert the movie into your presentation.

7. Provenance

From the eSiMon dashboard, it is possible to view the raw data files used to create images and movies. This is an action that you as a user, usually would not be concerned with or that you should not have to be burdened with. One of the main advantages of using eSiMon is to hide these lower level details and allow scientists to focus on the sciences. In this tutorial we mention this feature as it is used throughout the eSiMon dashboard to link tree variables and movies/images, to download raw data file (next section) and for analysis.

a. Make sure the first cell is selected and **right-click** on the Status Bar
b. **Select Show provenance** from the right-click menu

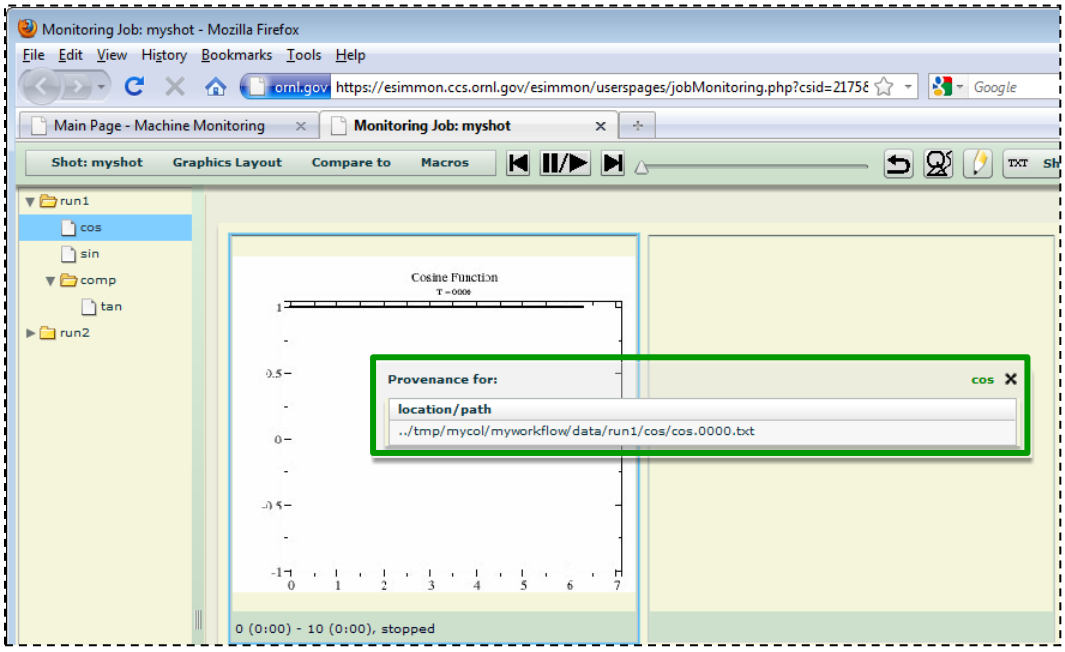

Figure 15: Provenance

8. Simulation Notes

You can take notes about this simulation.

a. **Click** on the "**TXT Show**" button on the Application Control Bar
b. A text area appears at the bottom of the page.
c. Take some notes about this simulation… and **click** on the "**Save**" button
d. **Click** on the same button on the Application Control Bar, now labeled "**TXT Hide**"
e. Your notes will be saved and retrieved any time you open this shot.
f. Optional: You can close and reopen the same shot from the "Search Collaborators' Old" from the Main Page or simply refresh the page to check that your notes have been saved.
g. Optional: If you reopened the shot, or refreshed, you will be back on Space 1, go back to Space 4 using the Graphics Layout Menu or continue this tutorial using variables and movies of your choice on Space 1.

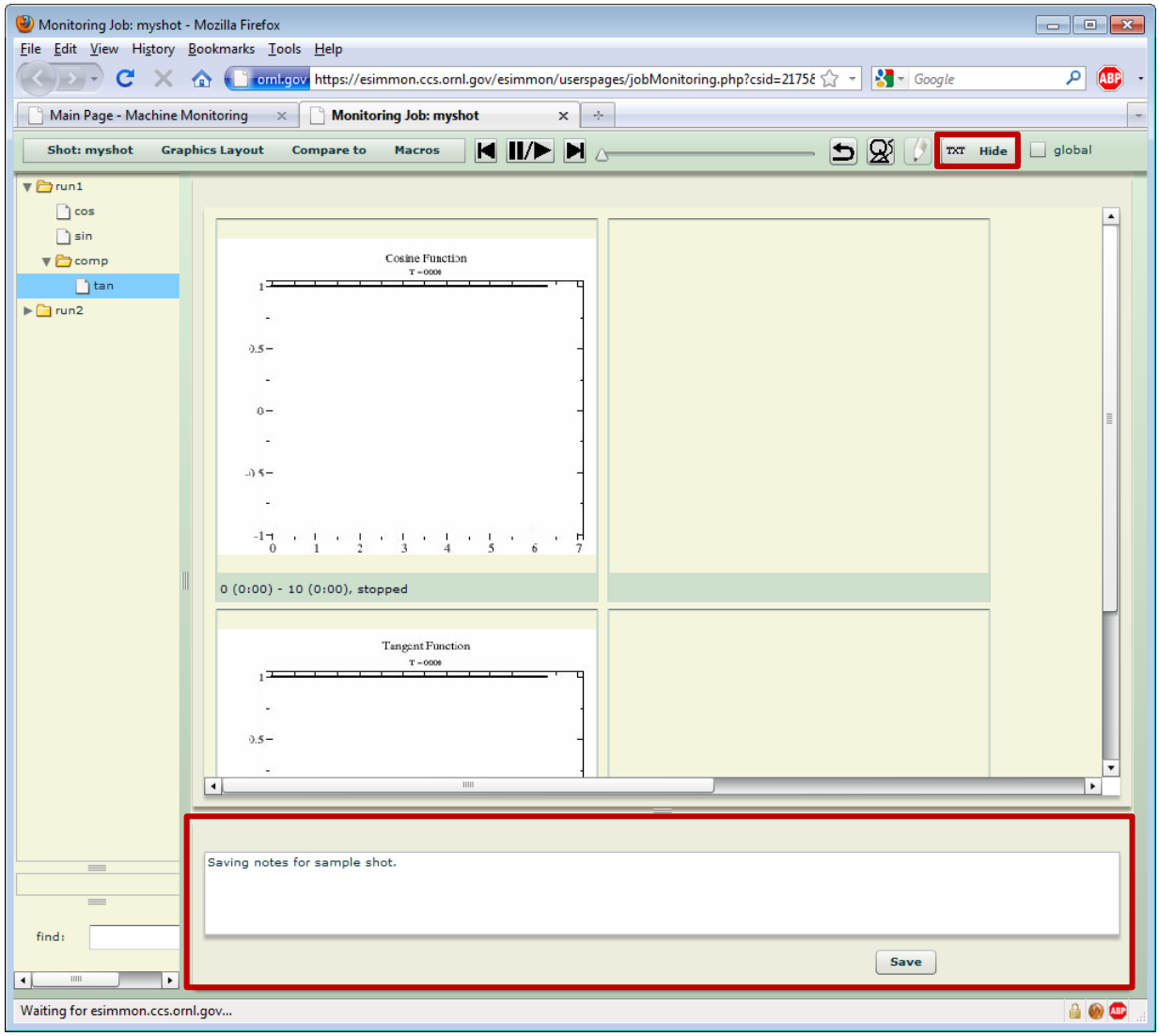

Figure 16: Simulation Notes

9. Movie Annotations

It is possible to make annotations on movies. To do so:

a. Select the "***cos***" movie
b. Click on the "**Annotate**" button (button with a pencil icon)
c. A new pop-up window appears
d. To make a new annotation:
    i. Select the type of annotations (**shape, text, free hand** or **arrow**)
    ii. Select whether this annotation should be for this particular time step or all time steps
    iii. Set the corresponding parameters. For example for an ellipse, set the **line color** and **line width**. To add an annotation, you have two options:
    iv. To draw your own annotation and give it a custom width and height: **click** and **drag** on the movie. Release the mouse when you are satisfied with the size of your annotation.

iv. Or **adjust** the **width** and **height** parameters of the annotation in the pop-up window and **click once** on the selected movie.

v. You can always undo your adds using the **Undo Add** or **Erase All** buttons

vi. When you are ready to save your annotations, click on the **Save**, the **Close** buttons

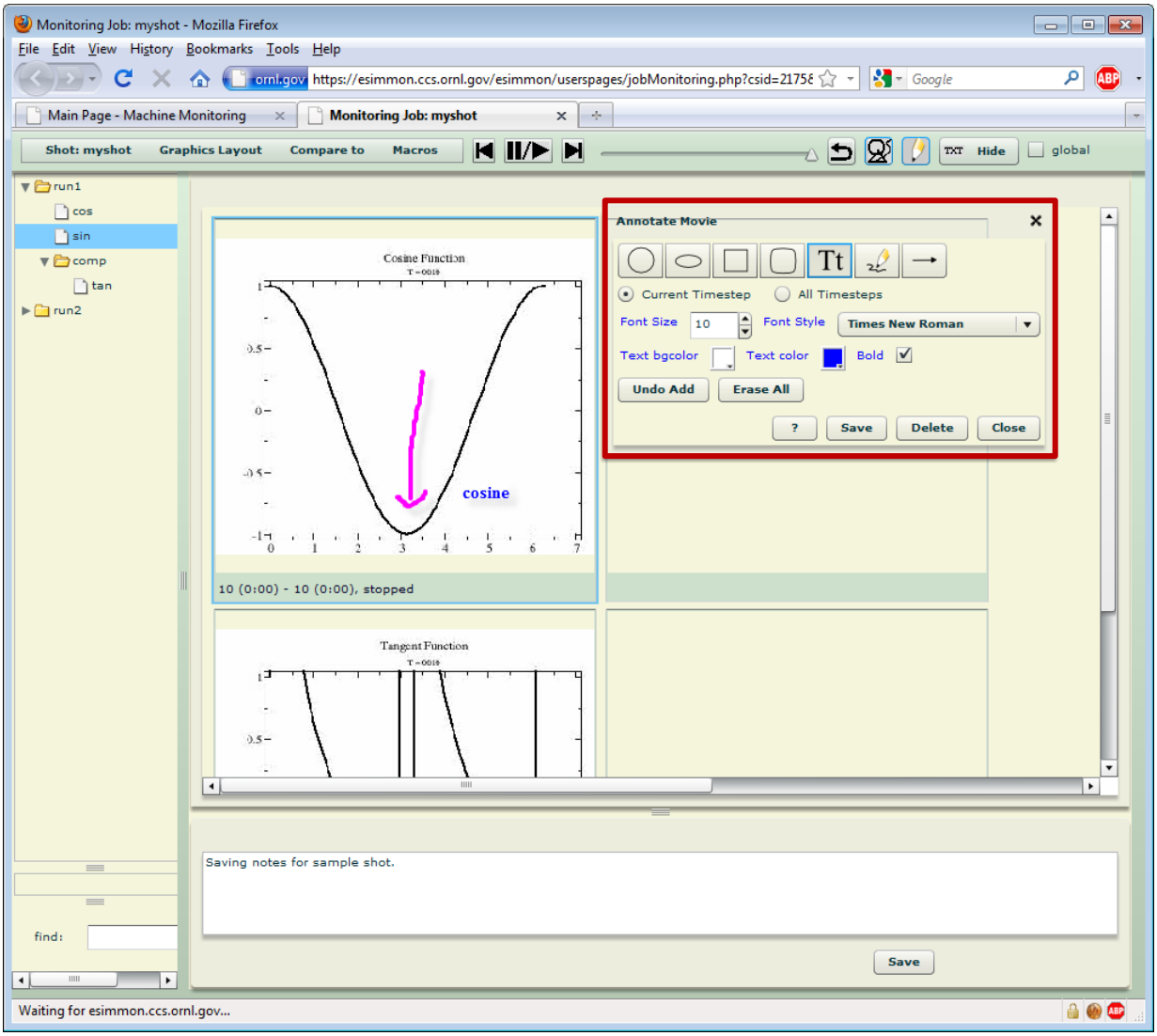

Figure 17: Annotations

Optional:

e. Once again to check that your annotations are saved. **Refresh** the page

f. **Navigate** your way back to Space 4.

g. Load Annotations by clicking on the "**Show Annotation**" (Annotations Icon) button next to the Annotate Button

h. **Play** Movies using the Play/Pause Button

i. You should see your annotation on the variable and time step you last saved it

10. Text Files

eSiMon also allows you to view text files (Figure 18).

a. **Open** the "***doc***" folder in the Tree View
b. **Start dragging** the "***doc***" tree item under the "**run1**" folder. The Video Cells in the Main Canvas are replaced by two Text Areas.
c. **Drop** the item onto one of the Text Areas
d. To **download** this a file to your client machine, **right-click** on either Text Area and select **Download** from the context menu
e. Similarly, select **Clear** from the Right-Click Menu to clear the Text Area. This is not necessary in order to load a new file. Simply drag and drop another item from the txt folder to view another file in the same Text Area

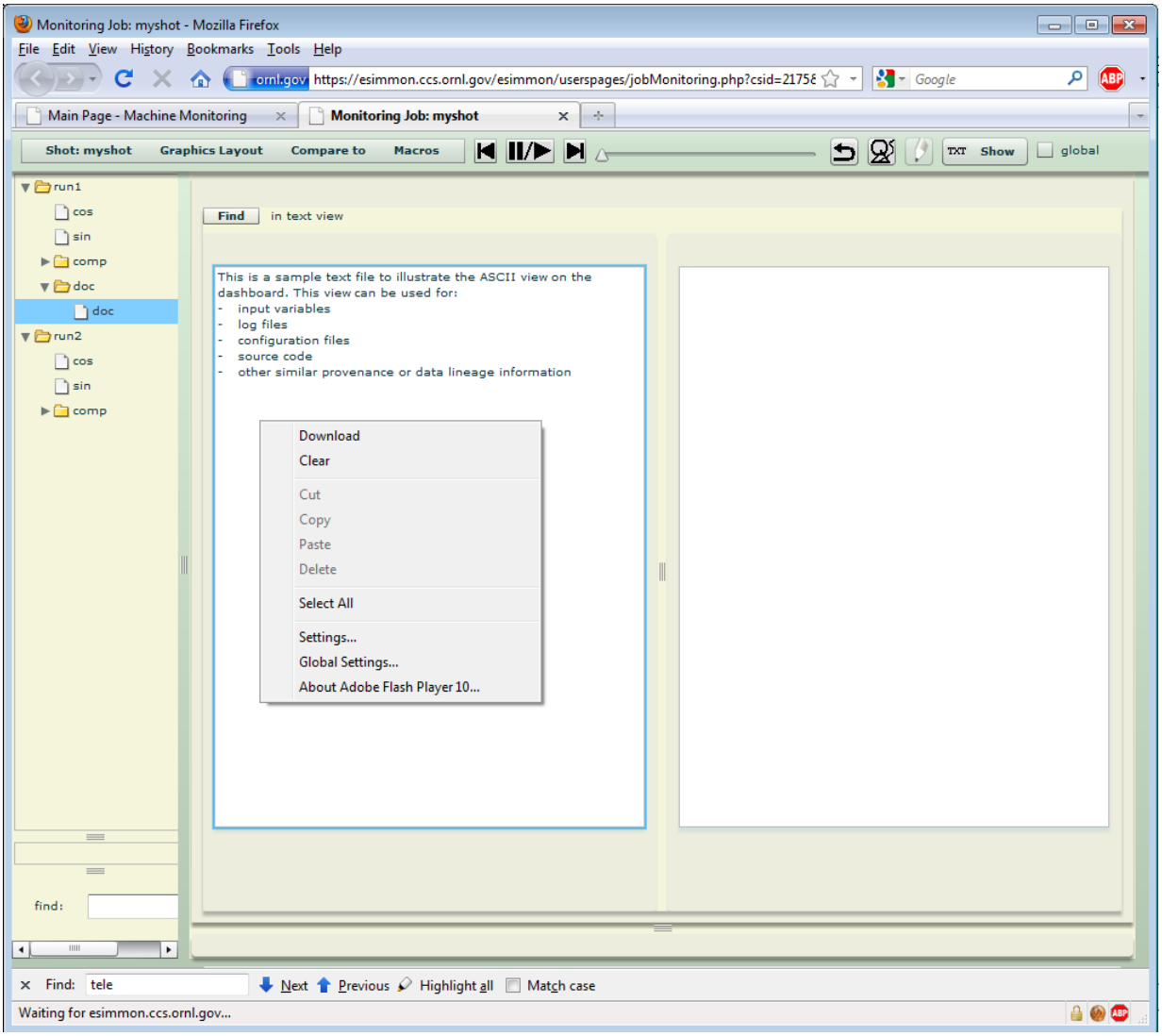

Figure 18: Text File View

You can return to the movie view by dragging initiating another drag from the tree or selecting **Graphics Layout** ➔ **Space** ➔ **ASCII view** from the Application Menu Bar.

E. Logout

You can logout from eSiMon from the Machine Monitoring page as well as from the Simulation Monitoring page.

From the Machine Monitoring page, **right click** on the page and select **Logout**. From the Simulation Monitoring, click on the ***shotname*** (first control) on the Application Control Bar and select **Logout**.

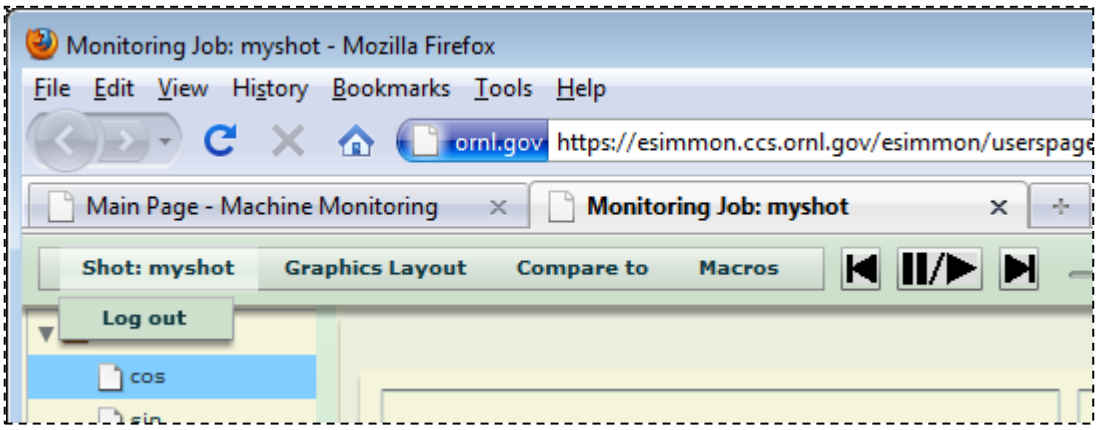

Figure 19: Logout from Simulation Monitoring Page

III. Analysis

The eSiMon dashboard allows for two main built-in analysis tools: Vector Graphics and the Calculator. The goal for future generation of eSiMon is to enable the use of varied analysis and visualization software packages through a plugin mechanism.

It is important to note that the current analysis tools are available for **NetCDF** and **BP** file format only. Vector graphics will work with ASCII data in addition to NetCDF and BP however the assumption is that there is one ASCII file per variable/time step combination. The calculator only works for NetCDF and BP files. In these cases eSiMon uses a tool to extract data for one variable at one time step from raw data files. The following examples assume that the user has opted to install the additional sample shots during installation.

A. Vector Graphics

Vector graphics are available for one dimensional data (1D or variables with x-y plots). The main advantage of working with vector graphics instead of movies is that movies are pre-processed and unalterable. With vector data, users can zoom in and take a closer look at parts of the data. Vector graphics of simulation variables are available in single and multi-cell windows. The multi-cell format allows the correlation of several variables in space and time which can also be very desirable.

To prepare for this section of the tutorial, open **shotad040** or **shotad041** from collaborator mycol's old job tab view. Open the tree view of variables and drag a couple of variables onto the main canvas. In the illustration below we use **ion__parallel_flow** and **ion__poloidal_flow** as sample variables.

1. Single-Cell Vector Graphics:

This feature is available through the right-click menu of each individual graphic cell as shown in Fig. 20.

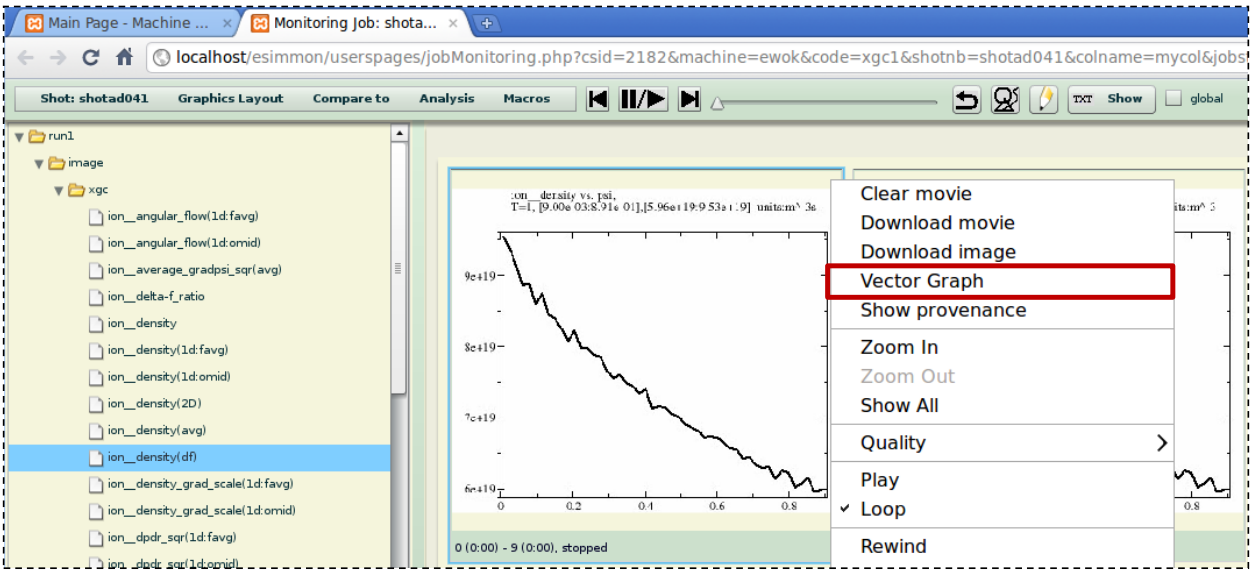

Figure 20: Vector Graphics Menu Item for a Single Cell

a. **Right-click** on the **ion__paralle_flow**
b. **Select** "***Vector Graph***" from the right-click menu. A new pop-up window appears on the screen
c. **Drag-and-drop** **ion__poloidal_flow** onto the same cell
d. Manipulate both graph using the control bar on top of the cell or the right-click menu of the pop-up window as indicated on Figure 21
e. Move to different time steps using the time navigation menu shown in Figure 21
f. Edit individual lines by **double-clicking** on the lines or symbols directly
g. Delete the second plot by **double-clicking** on it
h. **Drag-and-drop** a new variable onto the cell

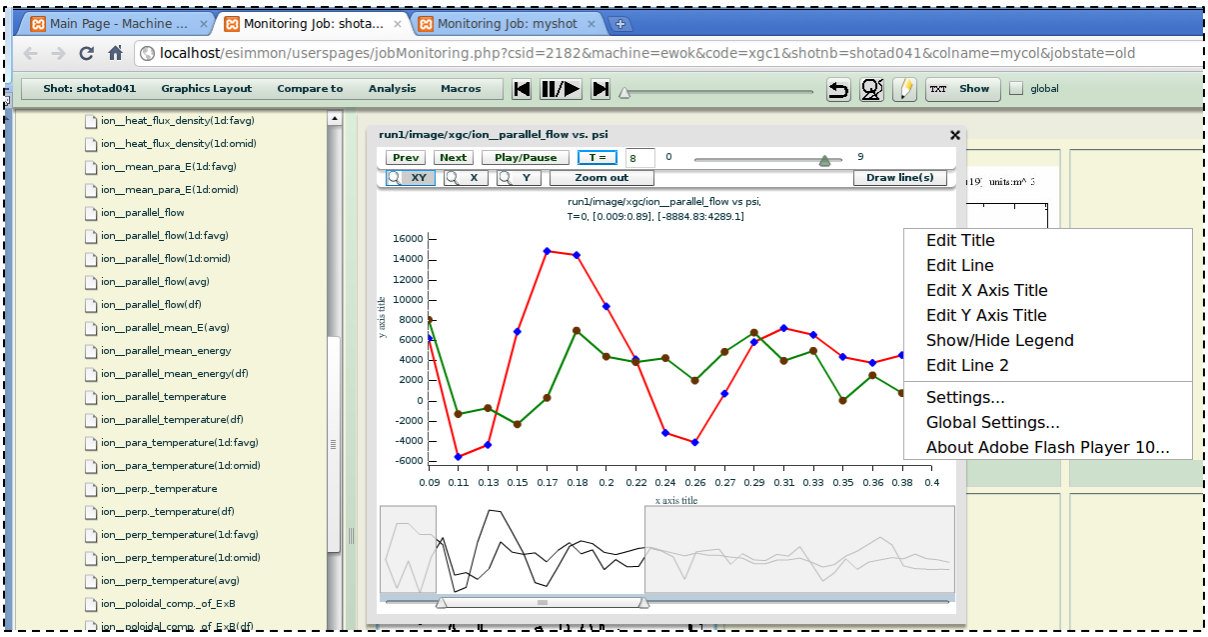

Figure 21: Single Cell Vector Graphics Window

1. Multi-Cell Vector Graphics:

The multi-cell vector graphics window is accessible through the Application Bar by clicking on the Analysis Menu Item, then Vector Graphics (Figure 22). A new pop-up window appears on the screen. The default number of cell is four.

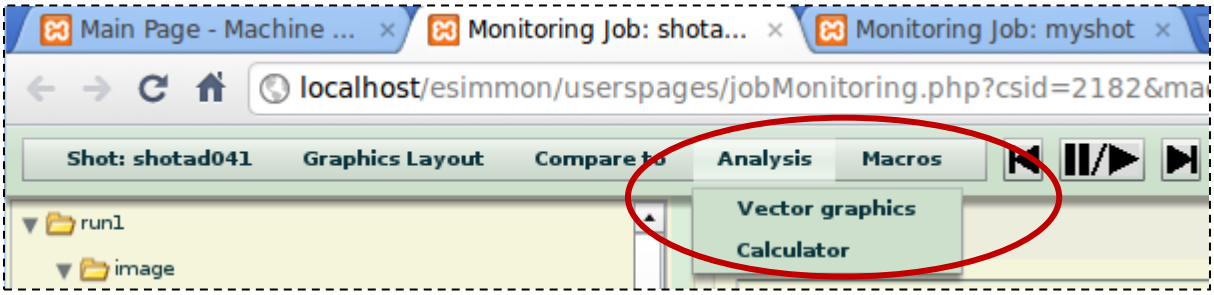

Figure 22: Analysis Menu

a. **Double-click** on the window upper border to maximize it
b. The default number of cell is four. Click on the "**Add Row**" button and the "**Add Column**" button once. You now have 9 cells total.
c. **Drag-and-drop** variables from the tree onto the different cells. Note vector graphics are not yet available for 2D variables
d. Manipulate graphs in the time and space dimensions using the control bar on top of the window or the slider at the bottom of the window

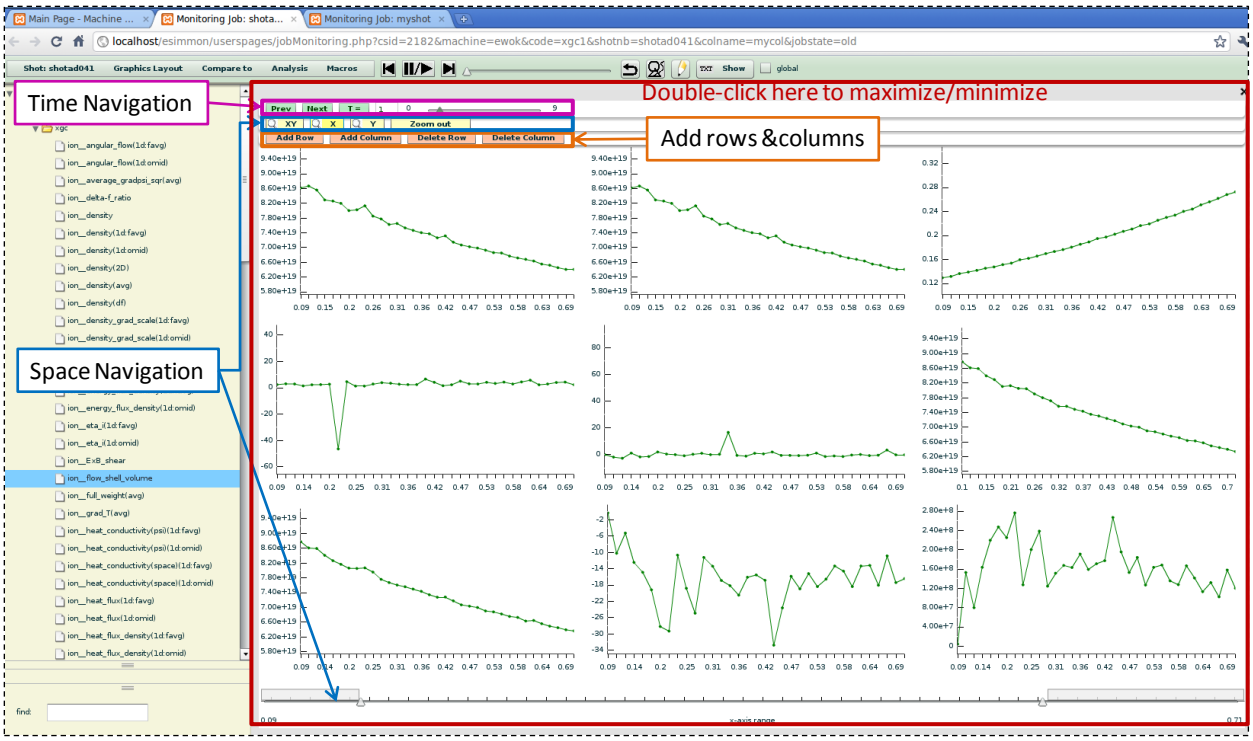

Figure 23: Multi-Cell Vector Graphics Window

B. Calculator

This Calculator allows basic operation between variables such as addition, subtraction etc. Its goal is to provide "quick-and-easy" exploration of the data before getting into more sophisticated analysis software and methods.

a. To view the calculator, **click** on the "**Analysis**" menu item from the Application Control Bar (see Figure 22) and select "**Calculator**"
b. **Select** variables by clicking on them. Make sure they are highlighted, then pressing the blue "**SLCT**" button on the calculator
c. **Click** on the **ion__parallel_flow** variable
d. **Click** on the "**SLCT**" button on the calculator
e. **Click** on the "**-**" operator
f. **Click** on the **ion__poloidal_flow** variable
g. Give a name to this operation by typing it in the "**analysis name**" input box (optional)
h. **Click** on the green "**=**" button. A new **analysis** folder appears on the tree view of variables
i. Wait for the clock cursor to disappear; **drag-and-drop** the newly created variable under the analysis folder onto a cell. See Figure 24
j. **Play** the movies
k. You can review which operation was ran by **double-clicking** on the variable on the tree. A new window appears on the screen as shown in Figure 24. This pop-up description becomes more and more useful as time goes by, the number of analysis increases and the names are no longer descriptive enough of the operation.

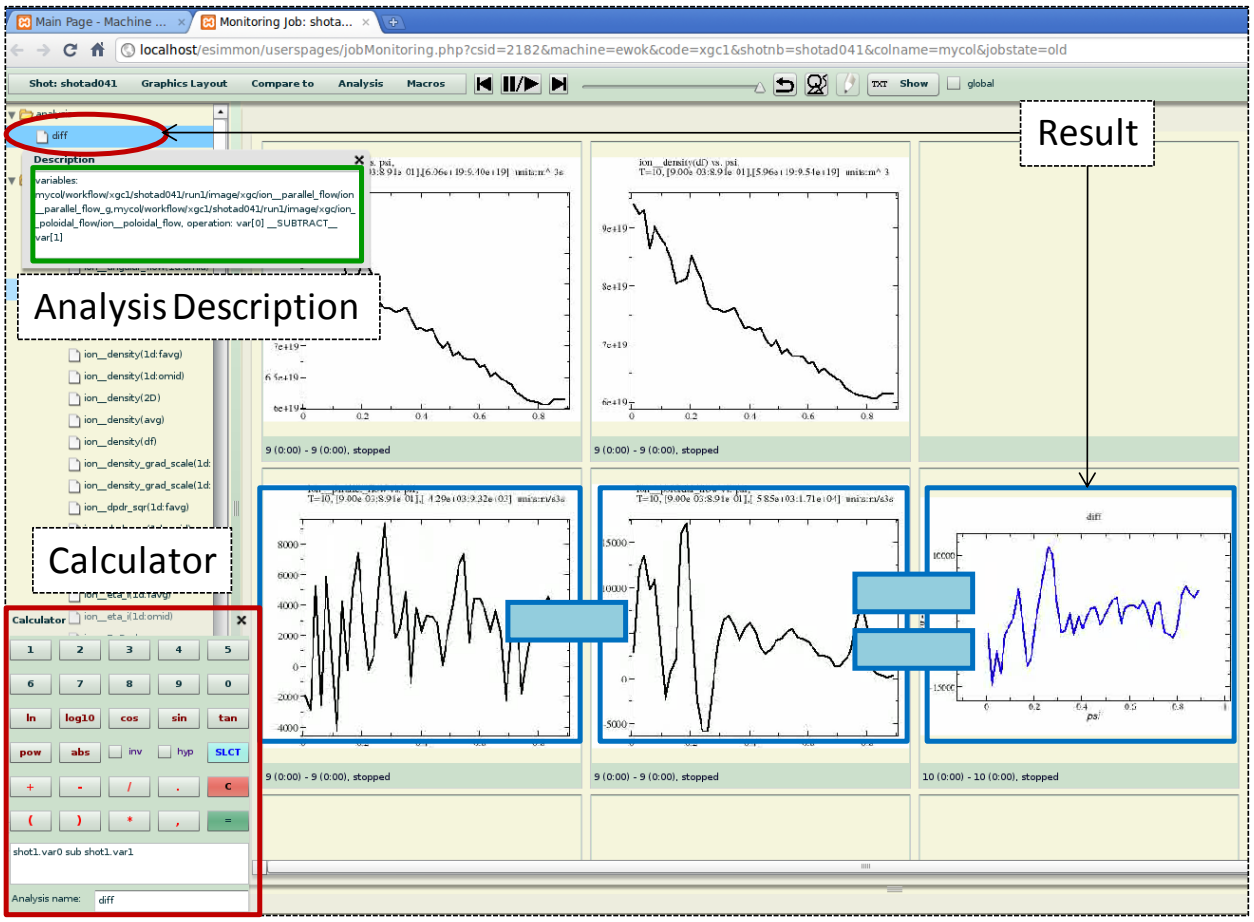

Figure 24: Calculator

IV. PHP API

We provide a PHP API with the eSiMon as a way to notify eSiMon of the initialization/finalization of a new run or the addition of a new variable for that run. This (PHP) API is mostly intended for registration of variables that are already visible and accessible by eSiMon. They also allow uploading new images and ASCII files to the simulation. It is possible to upload raw data to eSiMon as well. However if data size and upload speed are concerns, we will also provide a C API in the next eSiMon release.

A. Init.php

This API lets eSiMon know that a new simulation run has started. As mentioned earlier we call shot a simulation run. It has been our experience that scientists run the same simulation several time with very little differences in the input variables. For example, sometimes the reason for stopping and restarting a simulation is only the time they are allowed to run continuously on a system. eSiMon was designed with this concept in mind. Therefore one shot can have several runs. By default the first time users create a shot, it will always start with **run1**.

1. Input(s):

This file takes in 5 input variables: 3 required variables and 2 optional variables. The uniqueness of a run is based on the 3 required variables.

a. code name (PHP variable: **code** - required)
b. machine name (PHP variable : **machine** - required)
c. shot name (PHP variable : **shot** - required)
d. run name (PHP variable: **run** - optional)
e. shot directory (PHP variable **shotdir** - optional)

a. codename: The code name of the simulation being run. Some current and future features of the eSiMon dashboard are code based. Macros as defined in section II C 4 can be applied to different shots of the same code. For instance if there is a combination of variables that are useful to combine in general during a certain type of simulation, macros make it simple to correlate these same arrangement of variables in different shots.

b. machine name: The machine name where the code is being run. This variable is useful for identifying a single shot especially when a center has several computers available.

c. shot name: The shot name or number for one instance of the simulation being run.

d. run name: The run name or number is not required. By default it is "**run1**" for a new shot, unless otherwise specified by the user. There are two reserved values for the shot name: **"separate"** and **"continue"**.

- **seprate**: Passing "separate" creates a new run for a particular shot. In the case of the first run for a new shot, it will create "run1". If other runs already exist for the shot, it will create "runX" where x is the number of runs.
- **continue**: Passing "continue" will continue the latest run from a particular shot. In the case of a new shot, it will create "run1". If other runs already exist for the shot, it will set the status of the latest run to "running" to activate the running job monitoring on the eSiMon dashboard.
- string value: Passing a string name will create a run with that specified name under a specific shot. If such run already exists for the shot, it will set the status of that run to "running" to activate the running job monitoring on eSiMon.
  Passing s string is convenient for parameter studies for example. All the input to a simulation could stay the same except for one parameter and that could be represented using the run name.

e. shot directory: The directory where the data for the shot is located. If this value is not given, eSiMon creates a new shot directory under the directory specified in the configuration file. This directory must be writable by Apache.

Examples:

- **http://localhost/esimmon/api/init.php?code=mycode&machine=mymachine&shot=myshot**
- **http://localhost/esimmon/api/init.php?code=mycode&machine=mymachine&shot=myshot&run=param1**

- **http://localhost/esimmon/api/init.php?code=mycode&machine=mymachine&shot=myshot&run=param1&shotdir=/home/mycol/myworkflow/**

2. Output(s):

This file returns a ***cheatsheetid*** or a unique identifier for a shot. This number is passed on as a string or a number is passed as an input for any other operations regarding the monitoring and analysis of this shot.

B. Final.php

This API lets eSiMon know that a simulation is done running. This allows the transition between a running simulation viewer and a past simulation viewer on the eSiMon dashboard. It also creates FLV movies from all images under the shot directory.

1. Input(s):

The inputs for this file are either the *cheatsheetid* or the following grouped inputs: the machine name, the code, the shot name. In the case where the *cheatsheetid* is not provided, the run name is optional to set a specific run status to "done".

a. *cheatsheetid* (PHP variable: **csid** - required)
   or
a. code name (PHP variable **code** - required)
b. machine name (PHP variable: **machine** - required)
c. shot name or number (PHP variable: **shot** - required)
d. run name or number (PHP variable: **run** – optional). If this is not given, then any running shot from this shot will be finalized

2. Output(s):

Since, the creation of FLV and AVI movies happens at this time, filnaze.php returns either "**movie=succ**" or "**movie=fail**" to indicate whether this final step succeeded or failed.

Examples:

- **http://localhost/esimmon/api/final.php?csid=1234**
- **http://localhost/esimmon/api/final.php?code=mycode&shot=myshot&machine=mymachine**
- **http://localhost/esimmon/api/final.php?code=mycode&shot=myshot&machine=mymachine&run=param1**

Note: Users have the option of deleting jobs from eSiMon (see Appendix B). When using and testing the API, the *cheatsheetid* can be used to remove a run from the list of runs visible from eSimon. Simply run: **http://localhost/esimmon/php/deleteJob.php?csid=1234**

C. Add Data – addVariable.php

This API is used to register or upload/register a variable for a particular simulation on eSiMon. It informs the dashboard of a new variable for that simulation. In this particular case a variable is an **image**, a **plot**, a **graphical representation** of a physical entity. In other words, to view and monitor a variable on the web interface, eSiMon needs the simulation *cheatsheetid*, the name, path and type of the variable as well as the time stamp (or step) it corresponds to. Additionally, users can input the raw data file(s) from which these images were created.

Variables added using this API will appear on the **tree view of variable** displayed to the left of the simulation monitoring pages. When dragged and dropped from the tree to the main canvas cell, they will appear as graphics. There are 3 main ways of adding variables. The required and optional inputs vary according to these 3 distinct and mutually exclusive methods. If more inputs are given than necessary, the cases become blurred and the API returns an error. The *cheatsheetid* (PHP variable: csid) is required in all cases to associate the variable with a particular shot.

1. Register a variable (an image file) that already exists on the server:

In this case, the image files already exist on a disk accessible by eSiMon. The inputs needed are:

a. *cheatsheetid*: (PHP variable: **csid** - required)
b. directory path of the variable below the shot/run directory: (PHP variable: **filepath** - required)
   ex: If the variable full path is /home/scratch/mycol/mycode/myshot/run1/2d/density/density.0003.png where /home/scratch is the path to all shots and /home/scratch/mycol/mycode/myshot/ is the shot directory, then the path to the variable under the shot directory is simply **2d/density**. eSiMon will get the shot and run directory from the database using the *cheatsheetid*.
   If the full variable path as composed by eSiMon does not exist, registration fails.
c. image type: (PHP variable: **type** - required)
   ex: png, gif, jpeg, jpg
d. image time step (PHP variable: **tstep** - optional). This is an integer number representing the time stamp of the simulation. If it is given, eSiMon will generate a padded time step.
   ex: tstep = 5 ➔ timestep = "0005"
   Therefore a directory may include several files of type variable.xxxx.png for example. If the time step is not given, eSiMon will assume that the image file is overwritten at each time step and is not representing a variable that evolves over time. In that case, it will generate variable.png.
e. provenance file(s) (PHP variable: **prov** - optional). This is the full file path of the raw data file used to create the variable image file. If the image was created using several data files, the string is composed of their file paths separated by ";"

2. Input a variable (image data) to create a new image file on the server:

In this case, the API needs to create and register a new image file. eSiMon needs:

a. *cheatsheetid*: (PHP variable: **csid** - required)
b. file content: (PHP variable **filecontent** - required)
c. directory path under the shot/run directory: (PHP variable **filepath** - required). See example in section **III C 1 b**.
d. image type: (PHP variable **type** - required)
e. time step: (PHP variable **tstep** - optional). In not given, the assumption is that the file is overwritten at each time step and the filename is of the form density.png for example
f. provenance file(s) (PHP variable: **prov** - optional). This is the full file path of the raw data file used to create the variable image file. If the image was created using several data files, the string is composed of their file paths separated by ";"

Using these inputs, eSiMon will create a file such as variable.xxxx.ext or variable.ext under the eSiMon shot and run directories (see Appendix B) corresponding to the *cheatsheetid* and fill it with the content passed as the filecontent input.

3. Input a variable (an image file) to create a new image file on the server:

In this case, a file is being uploaded to the server. eSiMon needs:

a. *cheatsheetid*: (PHP variable: **csid** - required)
b. file: (PHP variable: **uploadedfile** - required)
c. directory path under the shot/run directory: (PHP variable **filepath** - optional). See example in section **III C 1 b**. If the file is not given, the path assumed is simply the filename of the uploaded file (white spaces are replaced by "_").
d. image time step (PHP variable:**tstep** - optional). If the time step is not given, eSiMon assumes the image file is overwritten at each time step
e. provenance file(s) (PHP variable: **prov** - optional). This is the full file path of the raw data file used to create the variable image file. If the image was created using several data files, the string is composed of their file paths separated by ";"

Using these inputs, eSiMon will uploaded the image file on the server under the eSiMon shot/run directories (See Appendix B) corresponding to the *cheatsheetid*.

Notes:

* During the monitoring, the simulation registers a variable at each time step. The information is the same each time except from the tstep input. Each time, the only value updated in the eSiMon database is this value. This is important when monitoring a running job. eSiMon displays the latest image in each variable directory and the greatest of tstep inputs for all variables of this particular shot. In contrast, if all the images files already exist for a past shot, the variable only needs to be registered once with the last time step.
* This API does not return anything when it is successful. In the case where a failure occurs, it aborts with an error message.

Examples:

- **http://localhost/esimmon/api/addVariable.php?csid=1234&filepath=comp/tan&type=png**
- **http://localhost/esimmon/api/addVariable.php?csid=1234&filepath=comp/tan&type=png&tstep=2&prov=/first/file/path;/second/file/path**

D. Add Data – addText.php

This API is similar to the previous the addVariable.php. It is used to register or upload/register a variable for a particular simulation on eSiMon. It informs the dashboard of a new variable for that simulation. In this particular case a variable is a **text file**. It could be an input file, a log file, a configuration file etc. In other words, variables added using this API will appear on **the tree view of variable** displayed to the left of the simulation monitoring pages. When dragged and dropped from the tree to the main canvas cell, they will appear as text. There are also three mutually exclusive ways to upload text on eSiMon and the *cheatsheetid* is always a required variable to identify the working shot. If the case is not clearly identified, registration fails.

1. Register a variable (a text file) that already exist on the server:

In this case, the image files already exist on a disk accessible by eSiMon. Inputs needed are:

a. *cheatsheetid*: (PHP variable: **csid** - required)
b. file path of the variable below the shot/run directory: (PHP variable: **filepath** - required)

ex: If the variable full path is /home/scratch/mycol/mycode/myshot/run1/log/logfile.txt where /home/scratch is the path to all shots and /home/scratch/mycol/mycode/myshot/ is the shot directory, then the path to the variable under the shot directory is **log/logfile.txt**. eSiMon will get the shot and run directory from the database using the *cheatsheetid*.
If the full variable path as composed by eSiMon does not exist, registration fails.

2. Input a variable (text) to create a new text file on the server:
In this case, the API needs to create and register a new image file. The inputs are:

a. *cheatsheetid*: (PHP variable: **csid** - required)
b. file content: (PHP variable **filecontent** - required)
c. file path under the shot/run directory: (PHP variable filepath - required). See example in section **III D 1 b**.

Using these inputs, eSiMon will create a text file under the eSiMon shot and run directories (see Appendix B) corresponding to the *cheatsheetid*.

3. Input a variable (a text file) to create a new text file on the server:
In this case, a file is being uploaded to the server. The inputs are:

a. *cheatsheetid*: (PHP variable: **csid** - required)
b. file: (PHP variable: **uploadedfile** - required)
c. file path under the shot/run directory: (PHP variable **filepath** - optional). See example in section **III D 1 b**. If the file is not given, the path assumed is simply the filename of the uploaded file (white spaces are replaced by "_").

Using these inputs, eSiMon will uploaded the text file on the server under the eSiMon shot/run directories (see Appendix B) corresponding to the *cheatsheetid*.

Notes:
* The text files are not associated with time steps.
* The filepath input is not just the directory of the variable, but the full path up to the name of the file. The extension may be omitted and the file will still be successfully displayed on eSiMon, but the input is the file path and not the directory path as it was in addVariable.php.
* This API does not return anything when it is successful. In the case where a failure occurs, it aborts with an error message.

Examples:
- **http://localhost/esimmon/api/addText.php?csid=1234&filepath=doc/doc.txt**
- **http://localhost/esimmon/api/addText.php?csid=1234&filepath=doc/doc.txt&filecontent=sometextcontent**

E. Add Data – addFile.php

This API differs from the two previous one in one key aspect. Data added using this API is not meant to be displayed on eSiMon. It consists of raw data files registered on eSimon for other purposes such as

performing analysis on the shot data. There are three similar ways to register/upload files on eSiMon. The *cheatsheetid* is always a required variable to identify the working shot.

1. Register a file that already exists on the server:

In this case, the image files already exist on a disk accessible by eSiMon. The inputs are:

a. *cheatsheetid*: (PHP variable: **csid** - required)
b. file path below the shot/run directory: (PHP variable: **filepath** - required)
   ex: If the variable full path is /home/scratch/mycol/mycode/myshot/run1/bg/xgc.bp where /home/scratch is the path to all shots and /home/scratch/mycol/mycode/myshot/ is the shot directory, then the path to the variable under the shot directory is **bp/xgc.bp**. eSiMon will get the shot and run directory from the database using the *cheatsheetid*.
   If the full variable path as composed by eSiMon does not exist, registration fails.

2. Input text to create a new data file on the server:

In this case, the API needs to create and register a new image file. eSiMon needs:

a. *cheatsheetid*: (PHP variable: **csid** - required)
b. file content: (PHP variable **filecontent** - required)
c. file path under the shot/run directory: (PHP variable **filepath** - required). See example in section **III E 1 b**.

Using these inputs, eSiMon will create a data file under the eSiMon shot and run directories (see Appendix B) corresponding to the *cheatsheetid*.

3. Input a data file to create a copy on the server:

In this case, a file is being uploaded to the server. The inputs are:

a. *cheatsheetid*: (PHP variable: **csid** - required)
b. file: (PHP variable: **uploadedfile** - required)
c. directory path under the shot/run directory: (PHP variable **filepath** - optional). See example in section III D 1 b. If the file is not given, the path assumed is simply the filename of the uploaded file (white spaces are replaced by "_").

Using these inputs, eSiMon will uploaded the data file on the server under the eSiMon shot/run directories (see Appendix B) corresponding to the *cheatsheetid*.

Notes:

* The files are not associated with time steps.
* The *filepath* input is not just the directory of the variable, but the full path up to the name of the file. The extension may be omitted and the content will still be successfully displayed on eSiMon, but the input is the file path and not the directory path as it was in addVariable.php.
* This API does not return anything when it is successful. In the case where a failure occurs, it aborts with an error message.

Examples:

- **http://localhost/esimmon/api/addFile.php?csid=1234&filepath=data/cos.000.txt**

V. C API

We provide a C API with eSiMon as a way to notify eSiMon of the initialization/finalization of a new run or the addition of a new variable for that run. It is possible to upload raw data to the eSiMon dashboard as well. The C API is defined in esimmon.h, following is a summary of the available interfaces.

A. esimmon_initRun()

This function initializes a run. It should be called only once during a simulation run. It takes the following input arguments:

1. **shot** (const char *): a name to identify a shot
2. **run** (const char *): a name for the run in the shot. RUN_SEPARATE is to be used to have separated runs in a shot (run1, run2, run3, ...). RUN_CONTINUE is to be used to continue a previous run (i.e. add new timesteps to that run and regenerate the movies).
3. **code** (const char *): a name for your code
4. **machine** (const char *): a name for the simulation host
5. **shotdir** (const char *): A directory accessible by eSiMon but created by the user.

This function returns a pointer to the connection of type **ESIMMON ***

You can put data/image/text here and register them without uploading data. The rundir field of the returned structure will be <shotdir>/<runname>, where runname is the specified <run> or what determined by the eSimon dashboard if using RUN_SEPARATE or RUN_CONTINUE. It is optional if the PHP method is used for uploading data. Then ESiMon will create a directory for you (on ESiMon's host).

B. esimmon_addVariable()

This function uploads and/or Register a variable (image). It takes the following arguments:

1. **connection** (ESIMMON *): the connection returned by esimmon_initRun().
2. **esimmon_path** (const char *): The relative path under the upload directory or connection->rundir. Should not include the image file name, just the path up to the parent directory (the parent dir name is the 'variable name').
3. **timestep** (int): A number as timestep.
4. **provenance** (const char *): A path to data file from which the image was created.
5. **mode** (enum ESIMMON_TRANSFERMODE): Image is already in place / upload a file now / upload from memory .
6. **data** (const char *): If the mode is file, then the data is local path to file, if the mode is memory, then the data is a pointer to data.
7. **content_length** (int): length of data in memory. If the mode is memory, give the size of data pointed by 'data'.
8. **imgtype** (const char *): if 'mode' is memory, the image file extension should be provided as well (e.g. "png" or "jpg")

This function returns an int. 0 if no error occurs, otherwise an error code.

C. esimmon_addText()

This function uploads and/or Register a text file. It takes the following arguments:

1. **connection** (ESIMMON *): the connection returned by esimmon_initRun().
2. **esimmon_path** (const char *): The relative path under the upload directory or connection->rundir.
3. **mode** (enum ESIMMON_TRANSFERMODE): file is already in place / upload a file now / upload from memory.
4. **data** (const char *): If the mode is file, then the data is local path to file, if the mode is memory, then the data is a pointer to data. If the mode is data, then data should be NULL and esimmon_path should be a relative path under connection->rundir.
5. **content_length** (int): length of data in memory. If the mode is memory, give the size of data pointed by 'data'.

This function returns an int. 0 if no error occurs, otherwise an error code.

D. esimmon_addFile()

This function uploads and/or Register a data file. Note that the File will not appear on ESiMon, but can be found/used by ESiMon in post processing actions, if variables have provenance pointing to such data files. This function takes the following arguments:

1. **connection** (ESIMMON *): the connection returned by esimmon_initRun().
2. **esimmon_path** (const char *): The relative path under the upload directory or connection->rundir.
3. **mode** (enum ESIMMON_TRANSFERMODE): file is already in place / upload a file now / upload from memory.
4. **data** (const char *): If the mode is file, then the data is local path to file, if the mode is memory, then the data is a pointer to data. If the mode is data, then data should be NULL and esimmon_path should be a relative path under connection->rundir.
5. **content_length** (int): length of data in memory. If the mode is memory, give the size of data pointed by 'data'.

This function returns an int. 0 if no error occurs, otherwise an error code.

E. esimmon _finalizeRun()

This function finalizes a run on eSiMon. This should be called only once per run. Note that after calling this function, the connection ID is terminated and no other calls will be allowed with that same connection ID. This function takes the following arguments:

1. **connection** (ESIMMON *): the connection returned by esimmon_initRun().

This function returns an int. 0 if no error occurs, otherwise an error code.

APPENDIX A

General Requirements for most Linux Based Operating Systems (OpenSuse, Ubuntu, Fedora)

Using the movie generating tools (MEncoder, FFmpeg) to create FLV movies from image formats other than PNG may require installing the appropriate libraries and header files. See MEncoder and FFmpeg documentation for supported encoders. Some of the following application dependencies will be found in default Linux distribution installations, others should be installed by an administrator.

General build tools	Movie generating scripts	FLVTool2	SRM-Lite	LessTif build
wget (or curl)	Python	Ruby	Java (Oracle/Sun Java recommended)	gawk
tar				bison
coreutils				flex (fast lexical analyser generator)
gcc				
gcc-c++ (sometimes packaged as g++)				
make				

Specific Requirements

Ubuntu

Tested on Ubuntu 10. Packages were installed using apt-get or Synaptic.

Ubuntu packages	Ubuntu 64-bit compatibility packages
libxrender-dev	gcc-multilib
libxt-dev	g++-multilib
libxmu-dev	ia32-libs
libxext-dev	
libxpm-dev	
x11proto-print-dev	

OpenSuSE

Tested on OpenSuSE 11. Packages were installed using Yast.

OpenSuSE packages	OpenSuSE 64-bit compatibility packages
	gcc-32bit
	gcc-c++-32bit
	xorg-x11- devel-32bit
	xorg-x11-libx11-devel-32bit
	xorg-x11-libXt-devel-32bit
	xorg-x11-libXext-devel-32bit
	xorg-x11-libXpm-devel-32bit
	xorg-x11-libXmu-devel-32bit
	xorg-x11-libXrender-devel-32bit
	xorg-x11-libXt-devel-32bit

CentOS/Fedora/RHEL

Tested on CenOS 5, Fedora 13 and Red Hat Enterprise Linux (RHEL) 5. Packages were installed using yum.

CentOS/ Fedora/RHEL Packages	Fedora 64-bit compatibility packages	CentOS/RHEL 64-bit compatibility packages
libX11-devel	compat-libstdc++-33	compat-libstdc++-33
libXext-devel	libstdc++.i686	libstdc++.i686
libXft-devel	libstdc++-devel.i686	libstdc++-devel.i686
libXpm-devel	glibc.i686	glibc.i686
libXrender-devel	glibc-devel.i686	glibc-devel.i386
libXp-devel	libgcc.i686	libgcc.i386
libXt-devel	libX11-devel.i686	libX11-devel.i386
	libXext-devel.i686,	libXext-devel.i386,
	libXft-devel.i686,	libXft-devel.i386,
	libXpm-devel.i686,	libXpm-devel.i386,
	libXrender-devel.i686	libXrender-devel.i386
	libXt-devel.i686	libXt-devel.i386
	libXp-devel.i686	libXp-devel.i386

APPENDIX B

Remove sample collaborator

If a user wishes to remove the example collaborator and sample shot from his or her eSiMon collaborators' tab view, his or she should navigate to the following url on a web browser:
http://localhost/esimmon/php/login/deleteSampleColShot.php

Users' home directories

An important assumption made in this software is that below the path to raw data, the shot directory always starts with the username.

ex: If the path to the raw data (**$rawData** in globalvars.php) in the configuration file is "**/home/scratch/**" all shot directories are assumed to follow the format "**/home/scratch/<username>**/…/…/…"

eSiMon directories

The eSiMon shot/run directories are first and foremost writable by Apache. It is specified as the **$dashbordData** in the PHP configuration file **golbalvars.php**. It's important and most times necessary to maintain separate directories for data created by users and data added through the web browser. Such directory is needed and used in the following cases:

1. When using the monitoring API, if the shotdir input is not set at initialization, eSiMon will create a shot directory where all subsequently uploaded data will reside. It creates a new directory under the **$dashboardData** directory.
2. **ex:** Assuming the path to eSiMon data is "**/tmp/scratch/**" (**$dashboardData**), eSiMon creates a new directory of the following format:
3. **/tmp/scratch** /**<username>**/**<codename>**/**<shotname>**

4. Even when the shot directory is set at initialization and is owned by users, data can be added to the shot using the PHP API or while doing analysis. Apache cannot write in the users' personal directories. Moreover sometimes users viewing a shot or doing analysis on a shot are not the owner of that shot, in which case they should not be allowed to write in the raw data directories. In these cases, a replica of the shot directory is created under the path specified in the configuration file.

   ex: Assuming the path to raw data (**$rawData**) is "**/home/scratch/**" and the path to eSiMon data is "**/tmp/ scratch/**" (**$dashboardData**), if the original shot directory is **/home/scratch/**mycol/mycode/myshot/, then the eSiMon shot directory will be **/tmp/scratch/**mycol/mycode/myshot/.

Removing runs

Each owner of a shot can delete a run from the “Old” and “Search Old” tabs by right-clicking on it. The “Delete Jobs” tab allows users to delete several runs at the same time.

This only removes the job from the user run listing on eSiMon. It does not delete any data. This is currently reserved for the users and/or administrators.

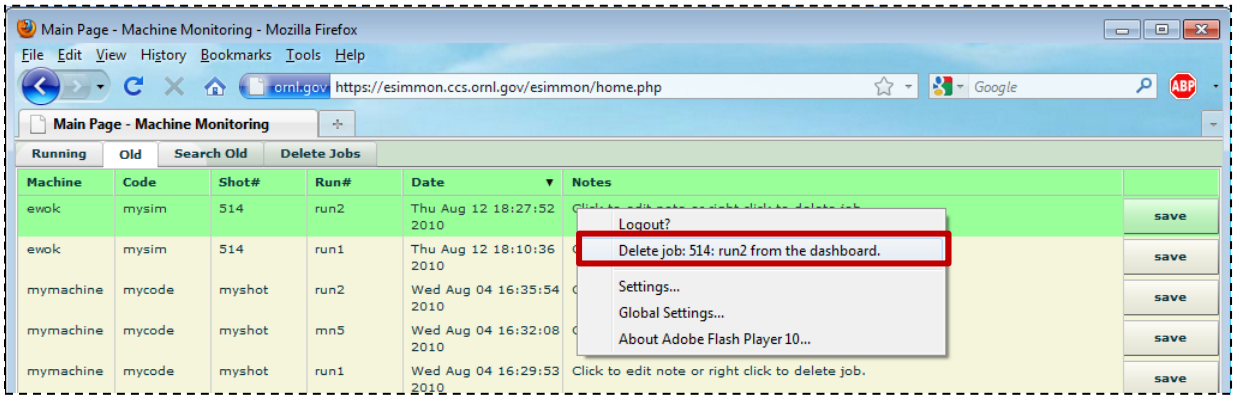

Figure 25: Deleting runs.