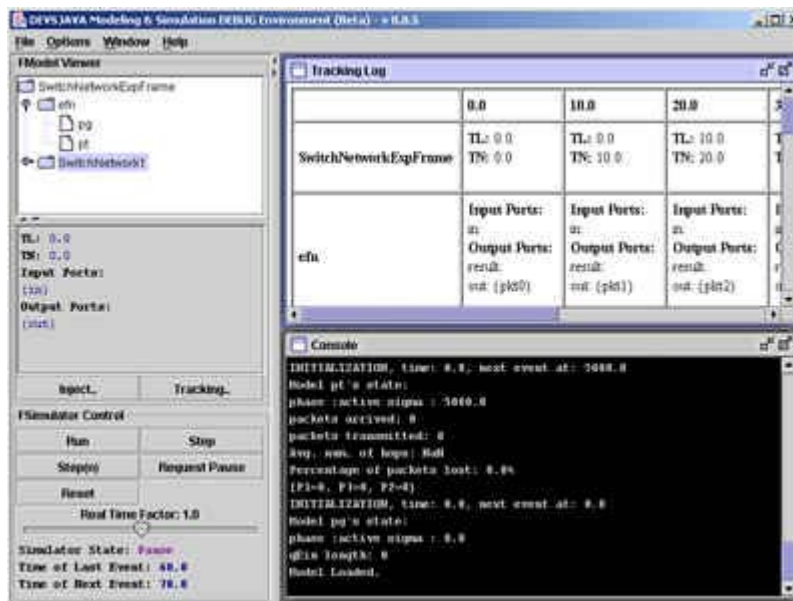


## DEVS Modeling & Simulation Tracking Environment User's Guide

### Table of Contents

- [Converting Models from the DEVSJAVA 2.7 Environment](#)
- [Loading a Model](#)
- [Reloading / Restarting a Model](#)
- [Tracking Model State Variables and Ports](#)
- [Saving and Exporting Tracking Information](#)
- [System Requirements / Performance Issues](#)
- [Known Issues](#)

The DEVS Modeling & Simulation Tracking Environment is a relatively simple tool originally designed to assist in the development of a larger, more sophisticated program. It has since grown a bit in functionality to accommodate the gathering of simulation data.



Note that the current Tracking Environment is in its initial release. As such, you are encouraged to report any bugs or any comments you may have. When reporting bugs with the software, please provide any errors that are given and (if possible) how to repeat the error. The email address to send to is: [Ranjit.Singh@asu.edu](mailto:Ranjit.Singh@asu.edu).

### Converting Models from the DEVSJAVA 2.7 Environment

To use models from the DEVSJAVA 2.7 Environment, several import and class-name changes need to be applied. The first step is to change the following import statements and class-names to a counterpart in the new environments'. (Note that Find/Replace mechanisms of certain IDE's can help with this procedure).

### Import Statements

SimView Import (old)	Debug Environment Import (new)
<code>import genDevs.modeling.*;</code>	<code>import devs.model.environment.modeling.*;</code>
<code>import genDevs.simulation.*;</code>	<code>import devs.model.environment.simulation.*;</code>
<code>import GenCol.*;</code>	<code>import devs.util.collection.*;</code>

### Class-Names

SimView Class Name (old)	Debug Environment Class Name (new)
<code>ViewableAtomic</code>	<code>atomic</code>
<code>ViewableDigraph</code>	<code>digraph</code>

Note that all other imports specific to SimView such as [`import simView.*`] should be removed or commented out.

The last step is to remove any GUI Information. SimView adds size and location information to the source file, this is usually found at the end of the file in the following method:

```
/**
 * Automatically generated by the SimView program.
 * Do not edit this manually, as such changes will get overwritten.
 */
public void layoutForSimView()
{
    preferredSize = new Dimension(834, 339);
    ((ViewableComponent)withName("SwitchNetwork1")).setPreferredLocation(new Point(349, 42));
    ((ViewableComponent)withName("efn")).setPreferredLocation(new Point(30, 22));
}
```

Please remove or comment out this method along with any other GUI specific code.

### Compiling Models From the Command Line

To compile model source code, the classpath variable must contain the Tracking Environment's jar file. For example -- If the jar file is located at {C:\TRACKING\_ENV.jar} and the source code is located at {C:\Models\simpArc}. A command prompt is opened and the current directory is set to where the source files are contained (in this case C:\Models\simpArc), when the compiler is invoked, the classpath is set to point to the jar file.

So here is what is entered after starting a command prompt:

```
C:\> cd Models\simpArc
```

```
C:\Models\simpArc> javac -classpath "C:\TRACKING_ENV.jar" *.java
```

**Note 1** - \*.java will compile everything in the current directory. This will produce the needed class files and the model is ready to be loaded.

**Note 2** - The quotes surrounding the classpath (C:\TRACKING\_ENV.jar) are required if the classpath contains any spaces.

**Note 3** - If javac is not available from the command line on your system, type in the following:

```
set path=%path%;<bin path> Where <bin path> is the path to the bin folder in the java installation directory.
```

**Note 4** - Once you know the list of commands to compile your models, you can simply save the list to a text file and rename the extension to [.bat], this way all you need to do is type in the name of the .bat file into the command prompt and it will automatically run all the commands again.

## Loading a Model

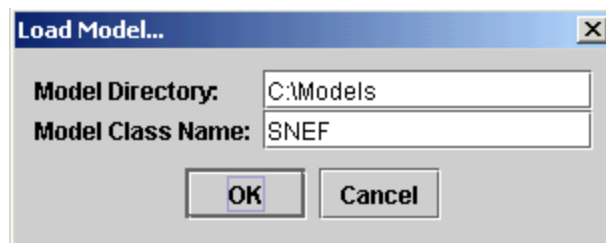
To load a model select [Load Model...] from the [File] menu. This will bring up a prompt asking for two pieces of information, the Model Directory, and the Model Class Name:

What to enter into these fields depends on if the model to load is part of a package or not.

### Loading Models that *are not* part of a package

To load a Coupled Model that is not part of a specific package, enter the *full path* to the folder that contains the model in the Model Directory field, and enter the class name of the Model in the Model Class Name field.

For example, if a Switched Network Experimental Frame (SNEF) Model is in a class called {SNEF.class} and is stored in the directory: {C:\Models}, then the fields will be filled in as:

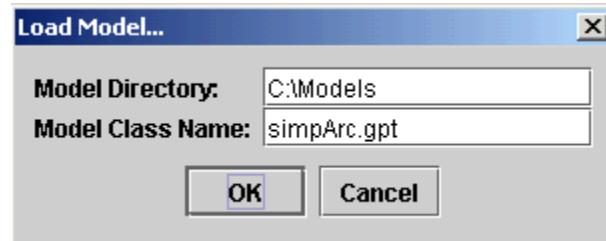


### Loading Models that *are* part of a package

To load a Coupled Model that is part of a specific package, the Model Directory field needs the *full path* to the directory that *contains* the model package. The second field, Model Class Name, is the *fully qualified name* of the model. Fully qualified names are the package name followed by a '.' (dot) followed by the class name.

For example, if a Generator-Processor-Transducer Model is in a class called {gpt.class} and

is part of the simpArc package located at {C:\Models\simpArc}, then the fields will be filled in as:



Notice that, in this case of packages, the Model Directory does *not* include the simpArc folder.

**Note 1** - Model Directory and Model Class Name are case sensitive.

**Note 2** - The debug environment can currently *load* only Coupled Models (digraph), but those coupled models may contain atomic models.

### Reloading / Restarting a Model

To restart a model, simply press [Restart] on the simulation control window (lower left corner) while the simulation is in an appropriate state. Note that restarting a model clears all tracked data including the console. To reload a model from its class file (helpful after a recompilation) select [Reload Model] from the [File] menu.

### Tracking Model State Variables and Ports

The tracking environment can track and record state and input/output information about a Model during a simulation. To select what data should be recorded, select a model from the FModel Viewer (top left corner) and click on the [Tracking..] button. This will bring up a dialog box presenting the possible State Variables / Input-Output Ports that can be recorded. Click on an variable/port to track it. Currently, the Model state variables that can be tracked are limited to phase and sigma, user defined variables are not supported.

As the simulation runs, the Tracking Log will contain a cumulative table of all model information being tracked. The Tracking Log can be viewed with the built in Tracking Viewer. By default the Tracking Log must be refreshed manually by selecting [Refresh Tracking Log] from the [Options] menu. Automatic refresh can be turned on, and the orientation (Horizontal vs. Vertical) of the Tracking Viewer can be set by selecting [Tracking Log Settings...] from the [Options] menu.

**Note 1** - Data is collected *post* external transition and *pre* internal transition for every simulation iteration.

### Saving and Exporting Tracking Information

To save the Tracking Log select [Save Tracking Log...] from the [File] menu. This produces a .html file with the Tracking Log table.

To export an individual model's tracked data to a Microsoft Excel™ readable format, select the model in the FModel Viewer and then select [Export to CSV] from the [File] Menu. This will produce a .csv (comma delimited) file for the selected Model that can be opened in Excel.

Some tracked data, such as phase and input/output ports, produce strings that are not easily manipulated in Excel. To solve this problem, the debug environment can *encode* string values into integer values. For instance if the phase column contains the strings Active and Passive, Active will become 0 and Passive will become 1. The algorithm guarantees each unique string to have a unique integer value (it is a 1 to 1 and onto function). Note that permutations are possible and are considered unique. For instance if an output port has the string "packet[0], packet[1]" at time 10, and "packet[1],packet[0]" at time 20, these are considered to be different strings. To export to an Excel readable *encoded* file, select a model in the FModel Viewer and select [Export to Encoded CSV] from the [File] Menu. This will produce two files, the first is a .csv file and the second is a .html file that contains a {key, value} decoding legend.

### **System Requirements / Performance Issues**

DEVS Modeling and Simulation Tracking Environment requires the Java(TM) SDK v1.3 or higher to run.

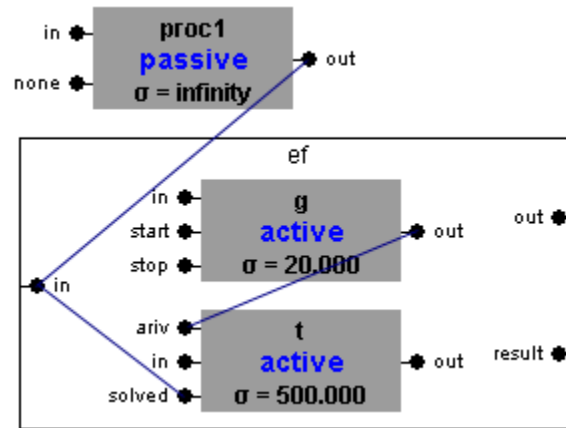
The Tracking Viewer uses Java's built-in HTML Pane for displaying the tracking log. This component is still in its infancy and is *highly* inefficient. For large tracking logs, expect a performance hit if the log is set to auto-refresh. When displaying large tracking logs with the built in viewer, it is recommended to have at least a 1.5Ghz machine with at least 256MB of RAM.

To avoid this performance issue, do not use the built in viewer for large tracking logs. Run the simulation, then save the log and open it with Internet Explorer or Netscape Navigator. Normal Requirements are any Pentium class CPU with 32MB of RAM or higher.

### **Known Issues**

To track an input port coupling that is connected to the input port of a coupled model, you must track and rely on the coupled models' input port, as the data being sent will not register on the atomic model.

For Example:



In this case, to monitor data being sent from the processor {proc1} to the transducer {t}, you can either track the output port {out} of {proc1} or the input port {in} of the experimental frame {ef}. The {solved} port of {t} does not register data. The {ariv} input port of {t} however, *will* correctly register all outputs from the generator {g}. It is only when data traverses through a coupled model, that it is not registered. (This is due to the nature of when data is recorded, it has no effect on the underlying simulator that still functions properly).

v 1.0.0