Olsonet Communications

SIDEMON

Version 0.1

January 19, 2014

1 Purpose

SIDEMON is a Tcl script for supervising multiple VUE[2] runs executed on the same "experiment server". The problem of running multiple instances of the emulator (and making sure that they are doing well) arose in the context of the Seawolf project. The idea is to use different agent sockets for the different runs, so the agents can reference them independently, safely, and consistently.

2 Running the script

The script can be run like this:

```
sidemon.tcl datafile logfile
```

where *datafile* is the input file describing the configuration of runs to monitor and *logfile* is the file to which the script will append log information. Both arguments are optional. If *datafile* is not specified, the script will try to open file *sidemon.xml* in the current directory. If *logfile* is not specified, there will be no log.

Note that the log is generally useful as it contains information about failures and errors. The log file is rotated by the script whenever it exceeds 5MB, with up to four previous files stored in files named as the original log file with suffixes .1 through .4.

The data file is in XML format. The best way to know what the script does is to understand how it interprets the data file.

An alternative way to invoke the script is:

```
sidemon.tcl -kill datafile logfile
```

with the interpretation of *datafile* and *logfile* as before. When called this way the script will terminate all runs described in the data file (if they happen to be alive).

Depending on the setting, you may be able to terminate the script once it has started all the required runs and the runs will remain alive (at least for as long as nothing goes wrong with them). Another option is to keep the runs associated with the script, such that they will disappear (be killed) automatically when the script is terminated. In the latter case, there is no need to resort to the "kill" version of the script call. In the former case, the script is idempotent, i.e., you may have multiple instances of the script running on the same data, and they won't be getting into into each other's way.

3 The data file

Here is a sample data file illustrating all the parameters and options:

```
<sidemon dirbase="/home/pawel/SOFTWARE/PICOS/Apps"
   delay="4" interval="15" sockbase="3072" detach="yes">
      <params> … </params>
      <side followproject="yes" socket="2011">
          <path>PROPRIETARY/eco_demo</path>
          <output>temporary.txt</output>
      </side>
      <side>
          <path>RFPing_u</path>
          <data>data8n.xml</data>
          <output append="yes">
              junk.txt
          </output>
      </side>
</sidemon>
```

The complete set is encapsulated into the single *<sidemon>* element. Its attributes are as follows:

dirbase This is the path to the base directory to be used as a prefix of all individual

	directories for the projects from which the runs will be taken. Normally, it makes sense to set it to the *App* directory of PICOS, or to the subdirectory of *App* covering all the relevant projects. The attribute is optional: when absent, all individual directories of the projects should be specified as absolute, or they will be interpreted relative to the current directory, i.e., the one in which the script has been called. Note that any absolute directory always stands for itself.
delay	The amount of time in seconds for which the script will wait for a response from the emulator before concluding that the run is dead. The default is 4 seconds.
interval	This is the check interval in seconds. Every so often the script will be polling the runs for signs of life and trying to restart them if no such signs are present.
sockbase	This is the starting number for socket ports to be assigned to those runs that do not explicitly request specific port numbers. The first of those runs will be assigned the port number specified as *sockbase*, the second that number + 1, and so on. There is no default for this attribute, i.e., it can only be absent if all runs request their port numbers explicitly (see below).
detach	If this attribute is set to *yes*, all runs will be detached from the script, i.e., they will be running as independent daemons and, in particular, they will not terminate when the script is killed. Otherwise, and also when the attribute is absent, killing the script will have the effect of killing all the runs that were started by it.

The role of the <*params*> element is to specify any exotic call arguments for the emulator that will be applied to all runs. Normally, it should be empty or absent altogether. For example, here is the way to make sure that all runs are executed in a slow motion mode:

```
<params>-s 4.0 -r 1000</params>
```

Note that those arguments apply to VUE^2 part of the argument set.

Every run is described by one <*side*> element. This element has two optional attributes:

followproject	If set to *yes*, this attribute means that the call arguments for the emulator should be determined from the PIP project configuration of the run. For example, this concerns the input data set, including its possible augmentation by node data snippets associated with the boards configured with the project. Any slow motion settings will be inherited from the project as well.
socket	If present, the attribute defines an explicit socket port to be assigned to the run. If absent, the socket number will be derived from *sockbase*, as described above.

A <*path*> element occurring within <*side*> specifies the path of the run's project, i.e., in combination with *dirbase* (see above) it points to the directory containing the run's executable, which must be called *side* or *side.exe* (under Cygwin). If there is no <*path*> (or the element's body is empty), the run's directory is solely determined by *dirbase*. Note that when *dirbase* is not specified as well, the executable will be sought in the current directory (the one in which the script is invoked). Note it is not necessarily a bad idea to have multiple runs involving the same executable and differentiated by the data (and output) files.

A <*data*> element specifies the name of the run's data file, which is interpreted relative to the run's directory (see above). Neither <*path*> nor <*data*> have any attributes.

Note that the first run in the sample data file (see above) has no <*data*> element. This is OK, because the run sets *followproject*, which tells the script to use the data file specified in the projects configuration file. Of course, in such a case, the configuration file must exist and it must specify an input data file. If there is a non-empty <*data*> element, it takes precedence over any data file configured with the project, regardless of the setting of *followproject*.

The <*output*> element is optional and, if present, specifies the output file for the run. If the *append* attribute of the element is *yes*, the new output will be appended to any existing output in the specified file. In that case, the emulator will be called such that its output is written to standard output, which will be redirected to the specified file. Otherwise, i.e., when the file is simply overwritten, it is passed to the emulator as a call argument.

If an output file is defined for the run, and the run fails (i.e., *sidemon* has to try to restart it), then the script will scan that file and present in the log the starting date/time of the last (failed) run and any error information that can be extracted from the file. This will work regardless whether the file is appended or overwritten.

4 Final comments

The script checks whether a run is alive by issuing an agent CLOCK request to which the emulator should properly respond, if it is alive. This request has no limit on the number of simultaneous connections, so the checks do not interfere with any agents that may try to connect to the emulator at the same time.

When called with -*kill*, the script goes through all the runs and issues a STOP request immediately disconnecting the socket. This has the effect of gracefully terminating the run.