WIND RIVER

Wind River® Simics®

APPLICATION NOTE

Simics Support

4.6

Revision 4081
Date 2012-11-16

Corporate Headquarters
Wind River
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.

Toll free (U.S.A.): 800-545-WIND
Telephone: 510-748-4100
Facsimile: 510-749-2010

For additional contact information, see the Wind River Web site:
`www.windriver.com`
For information on how to contact Customer Support, see:
`www.windriver.com/support`

Chapter 1

Simics Support

This document provides guidelines for interaction with Simics Support regarding inquiries or bug reports. The purpose of this document is to provide Simics users with the information necessary to improve the support process by decreasing the turn around time and increasing the overall quality of the support.

This document also contains useful information on how to extract information, pin-point errors and to debug the user's own work.

1.1 Glossary

This is some terms used to disambiguate our communication:

Host machine
The hardware you are running Simics on, your physical PC.

Host operating system
Linux or Windows run at your host machine.

Target machine or target system
The virtual machine or system simulated in Simics.

Target operating system
The operating system or firmware running on the target machine.

Target application
Software running on top of the target OS in the target machine.

User
All kinds of Simics users are referred to as *user*.

User module
Any software developed by the user to be used with Simics.

Chapter 2

The Support Forum

We refer all support cases to the on-line forum at `www.simics.net`. The forum allows for data and resources being easily accessed by users and support engineers.

The forum may be used at the users' discretion for all kinds of technical inquiries and at any level, from how-to questions to requests for technical advice on complex matters within the contracted scope of support. The forum is also used to report bugs or anomalies in Simics.

2.1 Posting at the Forums

The intention with this section is to hint on how to receive the shortest possible turn around time. We aim at providing users with the best possible support. However, the user can help to make this process smoother by adhering to these guidelines.

A user may post new topics at wish. While multiple trivial questions may be posted in the same thread, it is preferred that more complex questions and issues are posted in individual threads, which allows for clarity.

2.1.1 Elaborate on the Objective

Most often the response to an inquiry depends on the user's objective. Hence, the support team needs details such as a high or low level explanation of the objective to be able to provide tailor-made solutions or an adequate response.

2.1.2 Environment Details

Several environments are used to host Simics, both hardware and software, and some users use more than one system in parallel. Hence, include both a note on the host environment, the host hardware and host operating system, as well as a note on the target machine, the target operating system, and the target application if applicable. The CLI command **version -v** may be useful.

2.1.3 Detailed Query

Simics Support most often needs explicit details regarding setup, Simics version, start-script, user models, etc. The better detailed the original message the fewer iterations for explanations and complements. Include the commands, parameters, and flags that were used.

- A full copy & paste from terminals and consoles always gives the best details, in some cases a screen-shot is better.
- Log files are useful for tracing execution scenarios, see the CLI commands **[object.]log-level** and/or **log-setup**.

Before posting a new issue, increase the log-level, sometimes this provides possible reasons for the issue. See chapters User-Side Debugging and Reproduction of Bugs and Issues.

2.1.4 Attachments

It is possible to attach one file per message at the forum up to a maximum size of 100 MB. This is done by *editing* an existing message where the option to attach a file is found bottommost at that web page. Hence, first post a message, then edit that message. Post several messages in the same thread to attach several files, or attach one archived file. Files and archives of any common format are recognized.

Files larger than 100 MB can be uploaded to the FTP site `ftp.simics.net/incoming`, choose a unique file name. Announce the file name in the appropriate thread at the forum.

Chapter 3

User-Side Debugging

The cases of bugs that may appear may be a bug in a Simics model or in Simics itself, or, a bug in the user's models or software. This chapter suggests some debugging methods, and it briefly mentions some utilities that have proved to be useful.

Save time by quickly gathering data that can be included in the first report at the forum. Possibly, if the cause is within a user model it can be found and addressed even before the problem is posted at the forum.

3.1 Increased Log Level

Depending on the kind of error, increasing the log level for the entire simulation, or for only one or a few objects, may hint at what is going on just prior to the error. Commands of interest:

```
simics> [object.]log-level level
simics> ⟨object⟩.log-group ...
simics> log-setup ...
```

3.2 Logging Network Traffic

In the case networking is involved, most often captured network traffic provides useful pointers to the problem. Simics comes with capturing tools started with the CLI commands **pcapdump** or **tcpdump**. Moreover, Simics supports applications such as Ethereal and Wireshark which are started with the CLI command **ethereal**. An informative capture file can be attached to the post at the forum.

3.3 Simics Debug Symbols

Simics 4.0 and later is delivered with debugging symbols. In the case Simics crashes, at 32-bits Linux Simics tries to print a stack-trace. At 64-bits Linux Simics prints the current frame, and user modules can be compiled with the compilation flag `-fno-omit-frame-pointer` to

provide even more information. Copy & paste the output from the terminal to the forum post for complete details.

In some cases the stack-trace is scrambled which may be due to a corrupt stack which in turn may be caused by memory corruption. Such a memory corruption may have happened long before the error surfaces and then tools such as Valgrind are most valuable in tracking down the location of the original error (see the document *Debugging User-Developed Simics Modules*).

3.4 Debug the User Models

Debuggers such as GDB are the most valuable tools in pin-pointing errors. Such tools will not show Simics source code but will present the thread of execution while in the source code of user modules, given that they are compiled with debug info. Even if the error is not in the user model this kind of debugging may still provide information on parameter values and conditions that will help Simics Support in pin-pointing the cause.

One way to run Simics under the control of GDB is to start Simics and advance close to the interesting area of investigation. In Simics, execute the command **pid** to read Simics' process identity. Start GDB using the flag `--pid`, continue as usual.

Traditional brute-force `printf` debugging and the similar is not to dismiss while tracking the path of execution, states, and parameter values. Remember that Simics has a good logging mechanism, refer to the *Simics Reference Manual* for more details.

3.5 Non-Public User Models

It is not uncommon that users cannot share model source code or resources necessary to reproduce a bug or an issue. By sharing logs and other run-time information it is many times possible to track down what is happening and why. If that is not possible, try to reproduce the issue in a minimal example run on any of the models provided with the Simics installation, so that Simics Support can reproduce it on public models.

3.6 State Assertion

With Simics comes the module **state-assertion** which provides the possibility to compare two runs of Simics that are supposed to be equal. This feature are mainly used for two reasons: to verify checkpointability, and to find where the execution paths begin to differ after changes in target software or changes in user models.

In short, state-assertion is to run the same configuration in two Simics sessions and comparing the state of the two at specified intervals. It is the attributes of the observed objects that are compared and any attribute that differs will instantly alert the user.

More on how to use state-assertion is read in the document *Debugging User-Developed Simics Modules*.

Chapter 4

Reproduction of Bugs and Issues

Currently the most common time sink regarding support is probably to enable Simics Support to reproduce bugs and issues. Most often the support team must repeatedly ask for details on how to setup the simulation, which scripts to use, which commands to execute at CLI, which commands to use at the target console, details on what to watch out for at the target console or at CLI, which virtual time the particular issue happens at, etc. Hence these topics are discussed in detail with the goal to save users considerable time with getting Simics Support on track.

Generally it is a good idea to *always pretend that a new Simics Support engineer is handling the case*, one who is not familiar with the user's host environment nor the target environment, thus making each particular case as self-contained as possible, and linking to appropriate threads at the forum or to available documentation.

4.1 Start-up Scripts

For scripts provided with Simics we still need to know possible environment variables, parameters, and flags. Provide the exact start-up command.

If user scripts are used, provide the support team with a script that can be used to trigger the particular issue or bug. It may be a good idea to concatenate several scripts into one script.

Verify that the script is self-contained and that it does not rely on anything not available to Simics Support. Incomplete test cases is a common cause for delays.

4.2 CLI Commands

If commands have to be issued at CLI, include as many of them as possible in the start-up script, or contain them in another script loaded by **run-command-file**.

Sometimes it is not feasible to script CLI commands, then, to enable Simics Support to reproduce the problem, detailed step-by-step instructions is a must.

Most often the virtual time is a great marker for when a particular event is expected to happen or to be executed, hence, remember the command **ptime [-all]**.

Include relevant output from CLI in the post at the forum, it may serve as a verification that Simics Support sees the same output as users do.

4.3 Commands at Target Console

Any manual interaction with the target console breaks determinism, hence all input should be scripted using the command **input** ***string***, probably while using script branches and **wait-for-**$*$ ***string*** commands. These are best included in the scripts mentioned in the previous section.

Include relevant output from the target console in the post at the forum, it may serve as a verification that Simics Support sees the same output as users do.

4.4 Provide a Workspace

Sometimes it is feasible to provide a workspace with the user modules necessary to reproduce a problem. Simics Support welcomes users to archive a self-contained workspace with the folder `modules` and `targets`, possible user specific files, such as `module-user.mk`, and folders containing all necessary images, include files, etc. Still, CLI commands and commands at the target console are better scripted, as read in the previous sections.

Note: Archives larger than 100 MB can be uploaded with any unique name to the FTP site `ftp.simics.net/incoming`. Announce the file name in the appropriate thread at the forum.

4.5 Provide a Checkpoint

If the test case does not depend on user models and resources not available to Simics Support a checkpoint may be the best solution. Run the simulation close to an interesting point and write the checkpoint. Verify that the checkpoint can run in isolation, without the resources normally at hand. Archive it and upload it to the forum or the FTP site.

Also in this case the necessary CLI commands and commands at the target console are better scripted, as read in previous sections.

Chapter 5

Support Checklist

This chapter provides a short checklist to be used before posting an issue at the forum. The intention is to save valuable time for You and for Simics Support. Remember that incomplete test cases is a most common cause for delays.

1. Before Posting at the Forum
   (a) While multiple trivial questions may be posted in the same thread, it is preferred that more complex questions and issues are posted individually, which allows for clarity.
   (b) Increase the log level close to the error to find potential candidates for the bug and to gather logs.
   (c) Capture and analyse network traffic.
   (d) Find possible candidates for the crash in the stack trace or in the output at the console.
   (e) Debug user models using for example GDB or Valgrind. Even if a bug is not found, gather as much details as possible.
   (f) Use **state-assertion** to ensure checkpointability.
2. Posting at the Forum
   (a) Describe what your overall objective is and what you are trying to achieve.
   (b) Include full details about your host environment and Simics version; use the **version -v** command.
   (c) Provide as many details as possible, copy & paste is prefered.
   (d) Copy & paste possible stack traces or crash information, including the surrounding output at the console.
   (e) For non-public user models, provide as detailed logs as is allowed for. Provide a minimal test case.
3. Provide Means to Reproduce the Problems. Enable Simics Support to rapidly reproduce problems by any one or more of the applicable items listed below:

(a) Include the start-up scripts and details on the parameters and the flags that was used. Verify self-containedness.

(b) Prepare and include scripted CLI commands and necessary details on expected output.

(c) Prepare and include scripted input for the target console and interesting output from the console.

(d) Consider providing an archived self-contained workspace.

(e) Consider providing a self-contained checkpoint.