WIND RIVER

Wind River® Simics®

APPLICATION NOTE

Managing Simics Installations

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

Understanding Managed Simics Deployment Details and Options

This document is intended to supplement the information found in chapter 7 “Advanced Installation Options” of the *Simics Installation Guide* and assumes that the user is familiar with the concepts presented in the *Simics Installation Guide*.

This document will provide the IT department or Simulation Support Team with the necessary information to plan a Managed Simics deployment that includes locally produced Simics modules in a manner consistent with local IT policies and procedures.

Chapter 2

Terminology

In this document we will use the following terms:

`[simics]`
: The path to the directory in which Simics Packages are installed, defaults to `/opt/Simics/` on Linux host and `C:\Program Files\Simics\` or `C:\Program Files (x86)\Simics\` on Windows hosts.

`[workspace]`
: A directory under the end user's control where all user modification are created.

Simics Package
: A directory of files providing specific Simics functionality named to include a version number.

Simics Installer
: An executable identified as `Simics-pkg-<number>-<version>-<host type>` that installs a Simics Package. On Windows the installer is a `.exe` file, on Linux a `.tar` file that contains a shell script installation program and an encrypted compressed archive. In both cases when the installer is run a decryption key must be provided.

Managed Deployment
: A Simics installation managed by an IT department or Simulation Support Team where the end user does not run any Simics Installers.

Chapter 3

Review of Basic Installation

A Simics installation consists of a number of Simics Packages. Each Simics Package will install as a separate directory in `[simics]` named `<package name>-<version number>`. This installation style allows multiple versions of each package to be installed simultaneously. Once installed the installation directory can be made read only (by default it is read write for the administrator and read only for users).

3.1 Linux Details

Packages for 32 and 64 bit Linux may be installed in the same directory without conflict. If both host types are installed for any packages in an installation, all packages must have both host types installed for the installation to be considered supported. Both 32 and 64 bit Linux use the same installation script and thus both host types may be installed simultaneously and either host type may install either type of package.

3.2 Windows Details

On 32 bit systems the 32 bit Simics Installers for Windows will install in `C:\Program Files\Simics\`, while on 64 bit systems the 32 bit Simics Installers will install in `C:\Program Files (x86)\Simics\`, while the 64 bit Simics Installers will install in `C:\Program Files\Simics\`. Both 32 and 64 bit versions of Simics can be installed simultaneously without conflict as they will be in separate directories. While installing 32 and 64 bit versions of Simics in the same directory tree will not break Simics it is only supported in the context of a managed deployment as this will break the Start Menu short-cuts and uninstaller.

3.3 Workspace Details

When a user runs Simics a workspace directory will be created and all runtime file modifications are made in this workspace. Additionally all user modifications are expected to be made in the workspace.

3.4 Package Associations

Each Simics base package installation contains a file called `.package-list`, which lists the packages associated with that base package. When a certain Simics version is started, this package list determines the available target models, unless an override is done in an individual workspace as discussed below. During installation, reviewing and modifying this file might be necessary in order to create an appropriate set of packages for each Simics version. The package list is managed using the script **bin/addon-manager.[sh|bat]** in the Simics base package installation.

Chapter 4

Advanced Installation Details

4.1 Other File Modifications

In addition to the installation directories in `[simics]`, running an installation will result in a few other modifications to the system.

On Linux:

- A `~/.simics-installer/` directory will be created for the installing user, containing one text file per Simics version (4.4 or 4.6) listing the installation path and caching the decryption keys used during the installation.

On Windows:

- Installation keys and installation path will be cached in the Windows Registry.
- A Start Menu group "Simics" is created, containing one directory for each installed Simics Package.
- An entry in the installed programs list is created, allowing the "Add Remove Programs" or "Programs and Features" control panels to be used to uninstall Simics.

None of these installation files are used at runtime and can safely be deleted once installation is complete.

At runtime Simics will create several text files in either `~/.simics/<version>/` or `C:\Users\<username>\AppData\Roaming\Simics\<version>\` or `C:\Users and Settings\<username>\Application Data\Simics\<version>\` or similar depending on the host OS definition of a user's home directory. These files contain a log of recent commands issued to the Simics prompt, open files, open windows, and user preferences.

4.2 Removal or Uninstallation

Simics provides no uninstaller on Linux as **rm –rf [simics]/<Simics Package>** is considered trivial. On Windows a standard uninstaller is provided to comply with Windows application installation best practices. The installer simply removes the Simics Package directory and removes the entries from the Start Menu.

4.3 Relocating an Installation

Simics installations are completely relocatable, though any workspaces connected to the installation will need to be updated (by running the **workspace-setup** script). Assuming the entire `[simics]` directory is moved no modifications are necessary inside of the individual Simics Package directories as all paths are stored as relative locations in text files. No symlinks are used in the Simics installation.

4.4 Workspace Relocation

Workspaces can also be relocated, though an update will be necessary at the new location to ensure all paths are correct. This update is done by running the **workspace-setup** script.

Chapter 5

Workspace Installation Association Options

Every Simics Workspace is associated with exactly one Simics Base package. This association is created when:

- When **workspace-setup.bat** or **workspace-setup.py** is run.
- If the Simics GUI is started without a workspace it creates one using `workspace-setup.py`.
- When the "change-workspace" command is selected from the file menu in the GUI, `workspace-setup.py` is called on the new directory.

The association between workspace and installation is stored in the text file `[workspace]/.workspace-properties/workspace-paths`, which also contains the paths to the Simics Model Builder and Simics Extension Builder packages, and on Windows host the path to the compiler. These paths are used when building Simics modules to allow module source code to reference header files in the builder packages.

In addition to Simics Base each workspace is associated with a set of packages. This set is defined by the contents of `[workspace]/.package-list`; if this file does not exist (which is the default) the association is defined by reading the `.package-list` of the associated Simicsbase package. This is defined by the Simics-root entry in `[workspace]/.workspace-properties/workspace-paths`.

The contents of either `.package-list` file are defined by the **addon-manager** as documented in the *Installation Guide*. The Simics Model Builder and Simics Extension Builder definitions will be updated to match the paths found in `.package-list` when **workspace-setup** is rerun.

Chapter 6

Managed Deployment Options

6.1 Unmanaged Deployment

This is the default deployment that is documented in the installation guide. With this deployment option all users are provided with:

- The location of a package archive, either at `https://www.simics.net/` or some local repository.
- All the necessary keys to decrypt the packages the user needs, since decryption is part of the installation process.
- A license file.
- An expectation that they can install Simics themselves, possibly using the instructions found in the *Simics installation guide*.

This option has the advantage that it requires minimal resources of the IT or simulation support team, but provides no mechanism to ensure that all users are using the same version of packages and provides no notification to the user of updates.

6.2 Copied Deployment

This deployment takes advantage of the relocatability of a Simics installation by simply copying an unmanaged deployment to additional file systems. Generally a `.tar` or `.zip` archive of an installation, and optionally a workspace, is created and placed in a central location. Users then extract the installation onto a local file system.

If a workspace was included in the installation it will have to be updated once the archive is expanded, as it needs to update the path to the Simics installation on the local machine.

The installation itself can be put in any location, as long as the paths in the `.package-list` file in the base package are all relative. It is possible to manually insert absolute paths in this file, and you need to check for this before announcing that a Simics installation is ready to be copied.

6.3 Version/Tool Controlled Copied Deployment

This deployment is similar to a copied deployment in that it copies an installation to an additional file system, however it does this by simply checking the entire Simics installation into a version control or tool control system. There are obvious issues with storing binaries in version control systems, but Simics is no different from any other software in this respect.

This does ensure that the installation used by any user is the exact same that was used before. Reinstalling Simics from a known set of versions of packages should result in the same setup — but there might be manual changes to the installation or tweaks to package lists and similar that this approach catches.

6.4 Simple Multi-User Deployment

This deployment is also well documented in the *Simics Installation Guide*. With this option someone with administrative privileges installs Simics packages in a central read-only location and provides users with the path to **workspace-setup** in the installed base package, enabling users to configure their own workspace using the central installation as its associated Simics installation. The assumption is that users can run programs from a central shared installation, as is commonly enabled using NFS or other network file systems.

This option has the advantages that it

- requires little work from the IT or simulation support team
- provides the ability for IT or simulation support team to inform users of upgrades, or to automatically upgrade users, when new versions are installed (see below: Central Management of Workspaces).

This option has the disadvantage that it doesn't easily support different teams of users having different sets of packages associated with their workspace.

6.5 Multi-User Deployment with local .package-list

This deployment option takes advantage of `.package-list` files in workspaces. With this option the central read only installation becomes simply a collection of installed packages where the Simics base packages may or may not contain `.package-list` files. The `.package-list` file in the users workspaces can be managed either by putting the workspace under revision control, through the use of some software distribution tool, or through a modified version of **workspace-setup**.

6.6 Hybrid Multi-User Deployment

This is the most common deployment when a dedicated simulation support team is actively developing models and testing or certifying Simics installation versions before making them available to a broader user base. In this option users outside of the simulation support team operate as in a Simple Multi-User Deployment while the simulation support team

uses per-workspace `.package-list` files which are frequently managed by the simulation support team members individually.

6.7 Central Management of Workspaces

When using a multi-user deployment the IT or simulation support team has several options to control the version of Simics in use by the broad user base:

- Users can be informed of new installations via email instruction telling them to run **workspace-setup**.
- When workspaces are under revision control the master workspace can be updated with the expectation that the user will update periodically.
- A Simics-commands file can be created in the Simics base package. This is a collection of Simics commands that will be executed every time Simics is started.
  - One option is to simply inform users of the new version, for example the file could contain: 'print "A new version of Simics is available, please run `/net/opt/Simics/Simics-4.6.1/bin/workspace-setup.py`"' or equivalent on Windows.
  - A second option would be to actively update for the user. Simics commands can, with several levels of indirection, call a program on the host, for example **workspace-setup** in the new version and then run the CLI command **restart-simics** which will restart Simics and thus run the version from the new workspace.

Chapter 7

User Created Modules

When users write and build Simics modules the compiled modules end up in the user's workspace. There are several options for sharing modules with other users:

- The source code to the module can be copied from one user's workspace to another workspace where it can then be compiled.
- The binary objects can be copied from one user's workspace to where it can then be used directly if the host types match.
- A customer package can be created and installed as with any other package.

Chapter 8

Fully Managed Enterprise Deployment Example

This example assumes an enterprise where a Central IT department controls a shared filesystem and limits or controls the Software a user can install locally. Additionally a simulation support team for developing and supporting locally created Simics modules is assumed.

In this example deployment a number of Wind River provided Simics packages are installed in a shared read only location, a locally created customer created package or packages are also installed in the shared read only location, and some Simics Base package contains a `.package-list` file pointing to a set of packages approved and supported by the simulation support team. User's a given access to Simics through a preconfigured workspace distributed either by some software distribution tool or a revision control system.

Members of the simulation support team use revision control to collaborate on source code to locally created modules. When the modules are considered complete a customer created package is created and installed locally for testing. This requires that members of the simulation support team use local `.package-list` files. Once the package passes testing it is provided to IT for installation in the central location. When a new package version, be it customer created or Wind River supplied, is installed by IT it is not automatically registered with the Simics base installation until the simulation support team validates the combination of packages and versions.

Once the set of packages is validated using local `.package-list` files the Simulation support team creates a new `.package-list` file to be placed in the Simics Base package to publish the new modules to the broader user base. Users are informed of the new version of email and through updates to the preconfigured workspace.