Intel® Simics® Package Manager

Application version 1.14.0

Table of contents

Introduction

Terminology & Concepts
- Packages
- Projects
- Platform Manifest
- Package Bundles
- Package Repository
  - Local Repository
  - HTTPS (Generic) Repository
  - Artifactory* Repository

Prerequisites and System Requirements
- Minimum Supported Operating Systems
  - Linux Kernel and GLIBC Support
    - Linux Kernel and GLIBC Experimental Support
  - Windows Support
    - Windows Experimental Support

The Graphical User Interface
- Preparation state
- Platforms View
  - Use Eclipse* IDE for Intel Simics software with your platform project
  - Install from Manifest
  - Install from Repository
- My Projects
  - The project detail view
- Package Manager View
  - Installation status bar
  - Package details view
  - Installation confirmation view
- Addon view
- Settings
  - General settings
    - Package repositories
    - Platforms Repositories
    - Installation Directory

Default Project Path

Read only installation paths

Local Package Keys

Advanced Settings

Multi User Configuration

Cache

Proxy Settings

Path to Powershell (Windows only)

Temporary Directory

Help

Third Party Licenses

Limitations

Release notes

The Command Line Interface

Examples

Getting started

Up and running with Packages

Running the project from command line

Installing from a Virtual Platform

Batch Mode and the Command Line Interface

Command Line Interface Examples

Adding the CLI to your path

Add to PATH on linux (Bash shell)

Add to PATH on Windows

Configuring the CLI

install-dir

package-repos

ro-install-paths

decryption-key-files

proxy

Installing through the CLI

Installing a Package Bundle

Creating a project with your bundle

Install latest base package

Installing a specific base package version

Uninstalling a base package

Combining uninstalling and installing a base package ....
Installing/uninstalling all packages from a repository ....
Non-interactively ....

Introduction

The Intel® Simics® Package Manager (herein referred to as simply the package manager or the app) is an application which manages Intel Simics packages more efficiently by keeping track of what packages are installed and shows newer versions available from the configured package repositories (package shares). It helps perform bulk installations and uninstallations and is overall a more pleasing way of browsing and exploring packages available from different repositories. It comes with both a graphical user interface (GUI) and a command line interface (CLI).

The app is supported to run on both Linux* and Windows* operating systems. See Minimum Supported Operating Systems for more details.

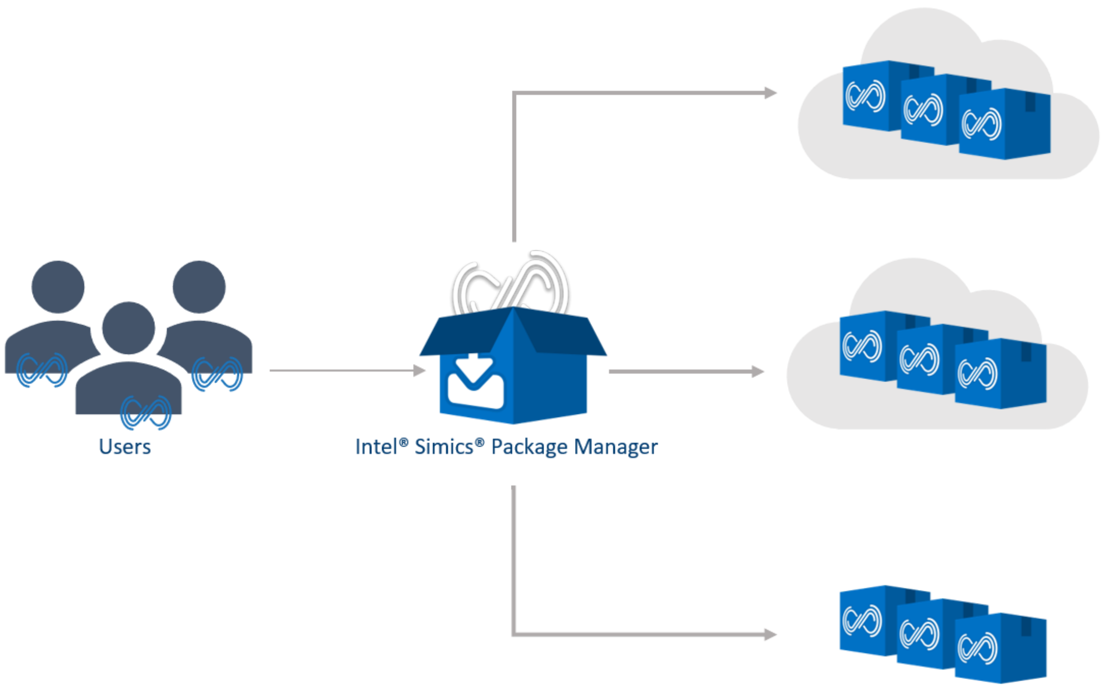


Figure 1 - Illustration of the installer connected to multiple repositories

The package manager will also manage new and existing Intel Simics projects. Import current projects or create new projects. Update projects when new packages are available.

For existing projects that depend on the addon manager, the addon manager has been built into the installer to simplify tracking what addons are associated with specific base versions. Using the addon manger is a requirement when working with projects created from the Eclipse* IDE for Intel Simics software.

A new concept called a Platform Manifest has also been introduced and will enable a bulk install of a specific set or recipe of packages (e.g. an Intel Simics Virtual Platform) while also providing descriptions and classifications to ease collaboration between teams. (See the Platform Manifest section)

Terminology & Concepts

This chapter lists terminology and concepts used in the package manager along with descriptions of their purpose.

Packages

An Intel Simics Package defines a collection of simulation models or software used to run simulations.

A notable component of this is the Simics Base package, which is the core software required to run additional models from other packages. Simics Base Package has the package number of 1000. The Express Package (package number 1003) is also considered a base package in this sense.

A base package is required in order to be able to run the simulator since it includes the simulator binary.

All other packages are considered addon packages. An addon package usually contains a subset of either tools or models that extends the base package. A virtual platform may depend on multiple addon packages along with a single base package.

Projects

A project is a workarea that allows you to configure the set of packages that should be used for a particular simulation. This makes it possible to have a different set of packages for different kind of works.

Platform Manifest

A Platform Manifest is a full description of a virtual platform with all the components and dependencies required to install, setup and run.

The manifest is used for making projects and platform recipes reprodicible in a quick way.

The purpose of the Platform Manifest is to empower users to share their virtual platform and know others can achieve the exact same setup elsewhere.

The package manager currently only supports fetching Intel Simics packages which mean that additional components that are not fetched during project setup (such as BIOS images or other firmware not part of the Intel Simics model being installed) need to be downloaded prior to creating the project.

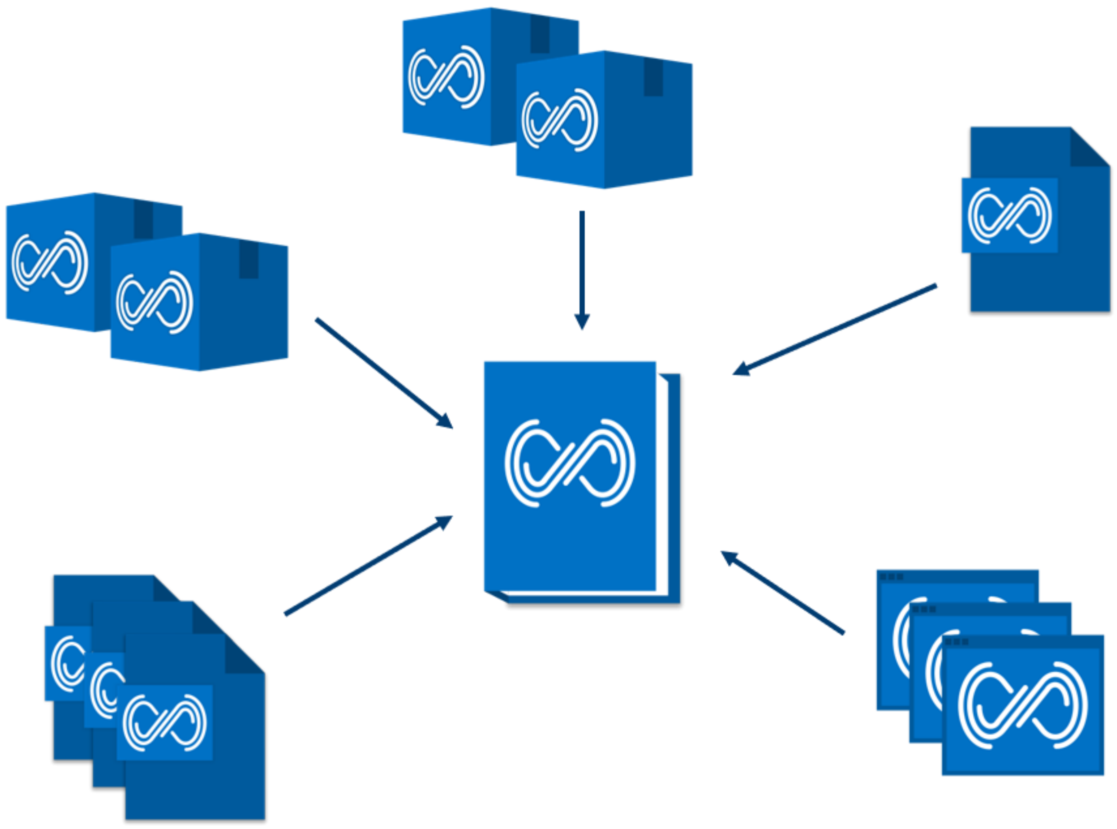


Figure 2 - Demonstration of content of a Platform Manifest

Package Bundles

A Package Bundle is an archive with packages that the the app can extract and install. Bundles are inferior to Platform Manifests since they are very big (gigabytes are to be expected) and that the app can't choose from where to download each package and take advantage of Package repositories that are closer and faster.

A Package Bundle can be identified by its file extension `.ispm` which is the same as a single package.

Package Repository

In the package manager you have a few different types of repositories that you can utilize to fetch packages and platforms.

```
simics-pkg-<package number>-<package version>-<linux64|win64>.<tar|exe|ispm>
```

For all types of package repositories, the package manager will not search for packages recursively in a local repository, only packages located at the top level in the specified directory will get picked up.

The package manager supports these different repository types:

Local Repository

A local repository is a repository that is accessible within the file system such as local disks, NFS, or samba shares.

The package manager will parse any repository path as a local repository path provided it does not begin with 'https://'.

The package in the local repository need to be in this format for the the app to be able to locate it.

```
simics-pkg-<package number>-<package version>-<linux64|win64>.<tar|exe|ispm>
```

HTTPS (Generic) Repository

When connecting to an https repository the repository path must point to a html page (e.g. a simple Apache directory listing page) where each package is listed in the current format:

```
<a href="simics-pkg-<package number>-<package version>-<linux64|win64>.<tar|exe|i
anything</a>
```

Example:

```
<a href="simics-pkg-1000-6.0.1-win64.ispm">simics-pkg-1000-6.0.1-win64.ispm</a>
```

Artifactory* Repository

Artifactory* repositories may also be used as a source for packages & platforms. The app supports the default configuration of such repositories provided they expose the directory listing to the package or platforms repositories in a way that matches this format:

```
https://<server>/artifactory/<arbitrary path within repository>
```

It is in this way that the app can differentiate between a simple HTTPS repository and an Artifactory* repository.

Using an Artifactory* repository also can yield security benefits through more stringent authentication and entitlement restrictions.

Note that the Artifactory* repositories are also single level. Packages within nested directory structures will not get picked up by the app.

Prerequisites and System Requirements

The app is built on top of Node.Js*, Electron*, and Chromium* and therefore maintains requirements that are based directly on the requirements of these dependencies.

Minimum Supported Operating Systems

Our continuous integration testing is done on the following platforms and we expect full functionality on these OS versions. Other OS versions and kernel versions may work out of the box, and we welcome feedback regarding issues, but we cannot guarantee support for them.

- SLES 15.SP2 (GLIBC 2.26, kernel 5.3.18)
- Windows Server 2019 (1809)
- Windows 11 (22000.1455)

> NOTE! On older OS versions, the UI may only work when run without sandboxing enabled, like so: `./ispm-gui --no-sandbox`

On Linux, OpenSSL* is used to validate package signatures and a minimum OpenSSL version of 1.1.1c is required. It may work with older versions, but we do not guarantee it.

Linux Kernel and GLIBC Support

Operating System	Architectures	Versions	Notes
GNU/Linux	x64	kernel >= 4.181, glibc >= 2.28	e.g. Ubuntu 20.04, Debian 10, EL8

Linux Kernel and GLIBC Experimental Support

These GLIBC and kernel combinations should work, but the underlying Node.js ecosystem only supports these experimentally and could break without notice.

Operating System	Architectures	Versions	Notes
GNU/Linux	x64	kernel >= 3.10, glibc >= 2.17	e.g. SLES 15.sp2, EL7
GNU/Linux	x64	kernel >= 3.10, musl >= 1.1.19	e.g. Alpine 3.8

Windows Support

Operating System	Architectures	Versions	Notes
Windows	x64, x86	>= Windows 10/Server 2016	

Windows Experimental Support

Operating System	Architectures	Versions	Notes
Windows	x64, x86	>= Windows 8.1/Server 2012	

The Graphical User Interface

This chapter will go through each of the different views included in the app.

> **Warning:** Using the new installer on Windows can modify existing packages on the file system. Be cautious not to change packages if you are not sure about what you are doing. Changes might be irreversible.

Preparation state

The first time you open the installer you will be optionally prompted to enter a package repository path (depending on your system configuration) and an installation path. This is the location where the package manager will store downloaded and installed packages. It'll also check for preexisting packages. On Windows hosts the default installation path will be:

```
C:\Program Files\Simics
```

On Linux hosts, the default is blank, due to users' varying preferences and system setups.

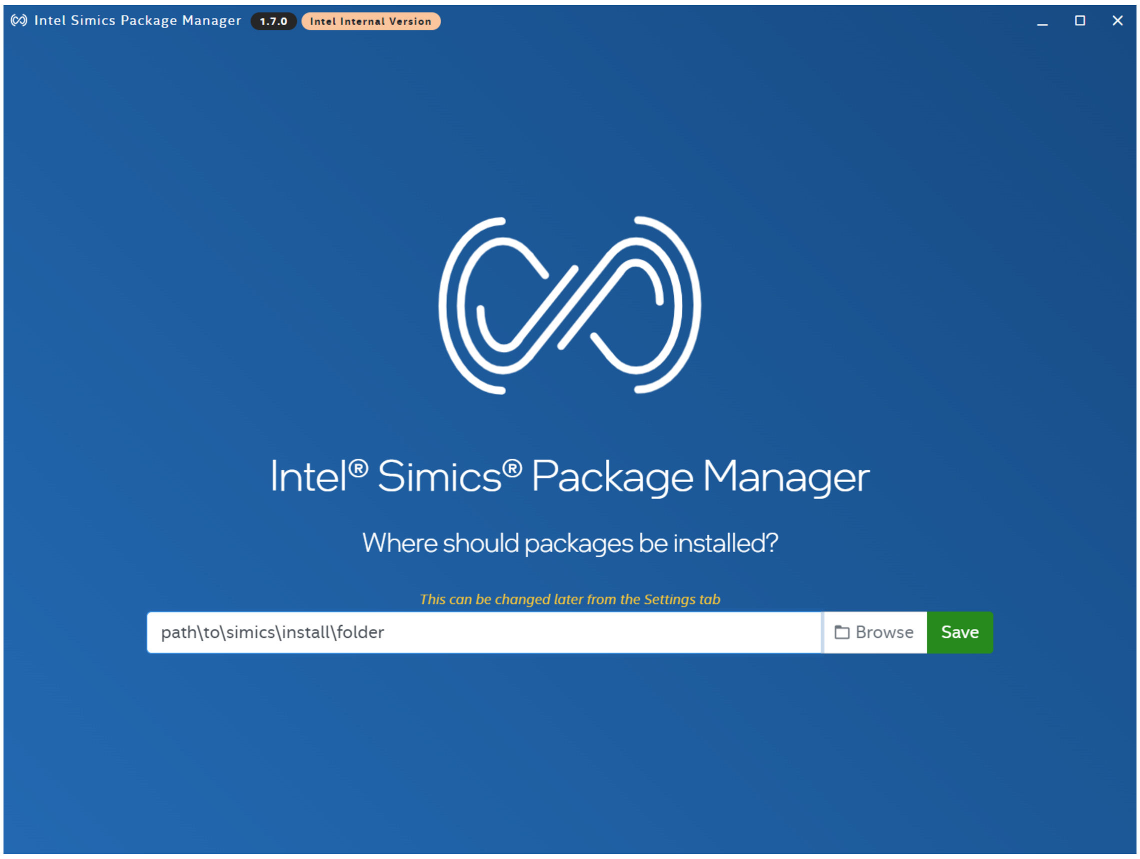

Intel Simics Package Manager 1.7.0 Intel Internal Version
Intel® Simics® Package Manager
Where should packages be installed?
This can be changed later from the Settings tab
path\to\simics\install\folder
Browse
Save

Figure 3 - Illustration of preparation state

It is possible to skip adding a installation path but then you will not be able to install any packages until you have added an installation path.

Platforms View

The Platforms view contains two sections:

- Install from Manifest
- Install from Repository

Intel Simics Virtual Platforms can be defined by a `.smf` manifest file matching a specific specification (contact support if you would like more information about this). Alternatively, a Platform can be defined by all packages contained in a single directory (locally or remotely on a package repository).

In the Platforms view you can select from a variety of different virtual platforms.

If no Platform repositories are available you can add platform repositories from Settings -> General

Selecting an individual platform will show you details regarding the platform. In the top right corner or underneath the platform title you will find a dropdown menu with different versions of the platforms. The default version is the latest available.

To create a project from a platform, click the **Create Project** button in the bottom right corner of the details view. This will create a project with default name and location. To customize its name and location, select **Create Project As** from the dropdown button. If you only need to install packages withoug creating a new project, select **Install Only**.

The Package manager will then show you what packages that will be installed and that your new project will be created. Press **proceed** and wait until everything has finished. After everything has finished if not done automatically navigate to the projects tab and verify that your project has been setup correctly.

Use Eclipse* IDE for Intel Simics software with your platform project

If Eclipse* IDE for Intel Simics software (package 1001) doesn't come bundled with your platform but you intend to use it, then you must follow these steps.

First of all, make sure you have package 1001 installed by navigating to the Packages tab. If you have a version of package 1001 already installed, you will probably see the package name in the sidebar instead of `Simics package 1001`. If it is not installed select the latest version by either checking the checkbox next to the package name or by Selecting a specific version in the package details which you reach by pressing the package name. From the details view you will be able to select what specific versions you want to use. When you have selected the version you want, press **Continue** and Install the package as previously described. Navigate back to the project page. Select your project. Press the **Manage Packages** link. Select package 1001 and then press **Update Project**. Your Project should now be setup and ready to be used with the Eclipse* IDE for Intel Simics software.

For starting the ide directly (e.g. from the Windows start menu) then package 1001 must be added to the desired base package's addon list. This can be accomplished on the Addons tab in the package manager.

Install from Manifest

The default view within Platforms is the **Install from Manifest** view. This view contains different sets of packages and files that work together in forming a virtual platform, all defined within a `.smf` file that lives in either a Platform Repository, or that has been locally imported using the add manifest folder button in the sidebar.

> **note:** A manifest does not have to contain a full virtual platform. It might only include packages that should be installed together for specific reproducibility.

When selecting a manifest, it will open in a detailed view that will contain:

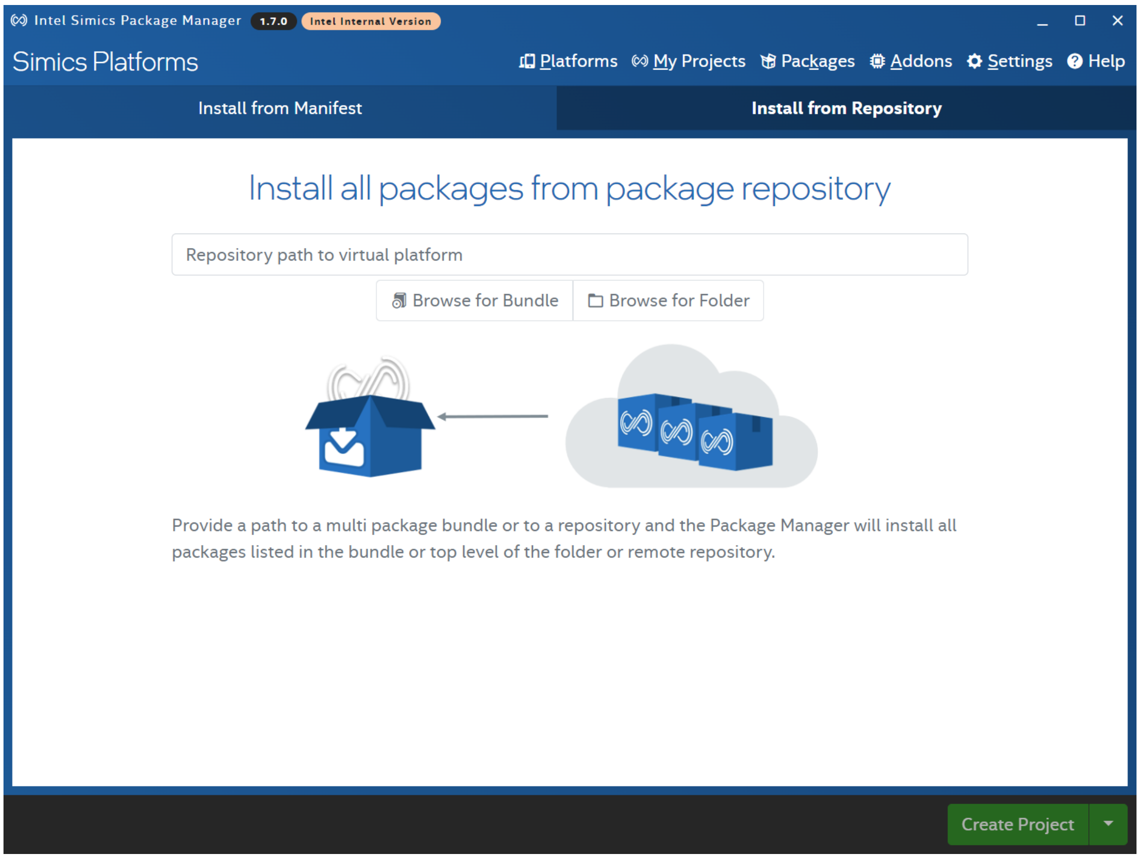
- Name of manifest.
- Description of manifest. *(if available)*
- Contact information.
- Packages included.
- Files included *(Not implemented in version 1.14.0)*

To create a project from a manifest press the **Create project** button when viewing the selected manifest.

You can manually import local manifests from your file system by pressing the folder button in the manifest sidebar. This will open a file dialog. The supported file extension is `.smf`.

Install from Repository

The secondary view in the Platforms view is **Install from Repository**



Provide a path to a multi package bundle or to a repository and the Package Manager will install all packages listed in the bundle or top level of the folder or remote repository.

Figure 5 - Illlustration of the Install from Repositroy view in the Platforms view

This view contains a text input that takes a path to a package repository or Package Bundle. If a repository is selected, this will find and install all packages at the top level of that package repository and is useful if you store packages grouped into platforms in a hierarchical folder structure.

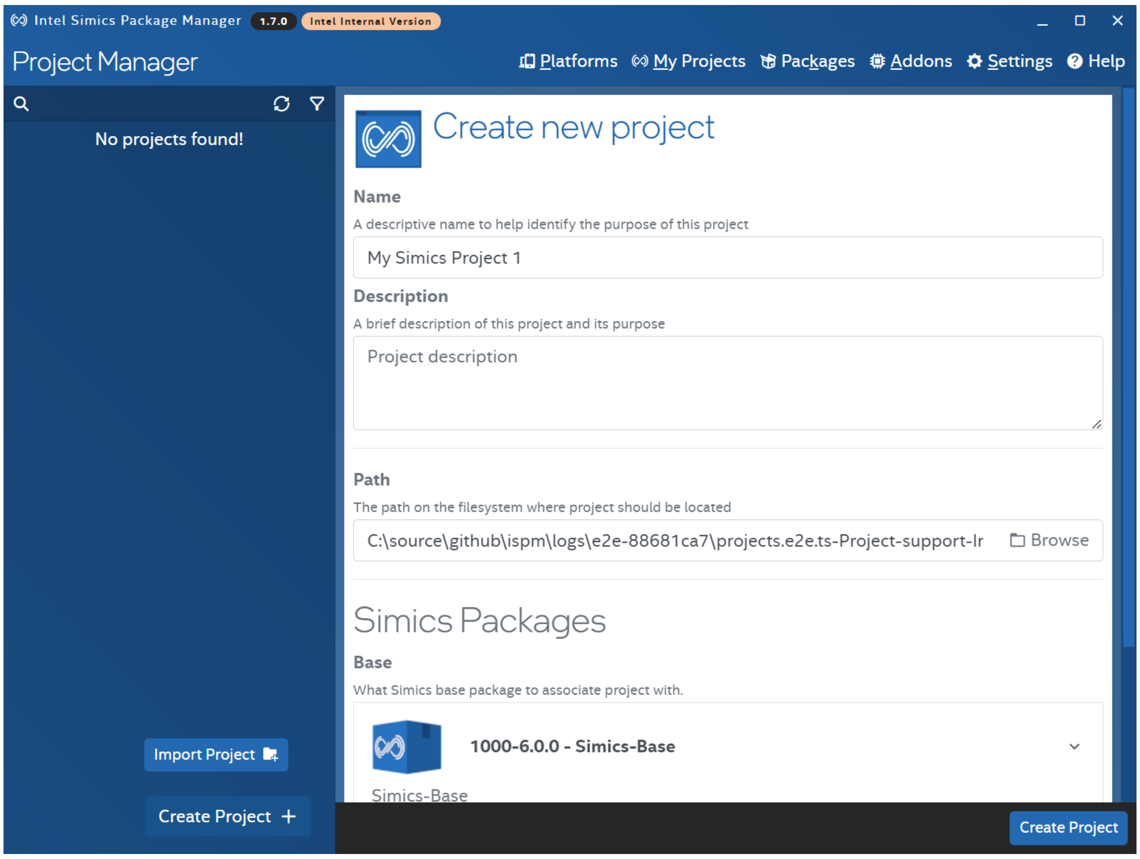
If a Package Bundle is selected the app will extract and install all the packages contained within the bundle.

My Projects

This view is intended to make is easier to create and manage projects.

If there are no projects in the installer yet, then the create project view will be shown.

Give the project a name and decide on where it should be created. Then select what base package to use and check what addon packages to associate.



Create new project

Name

A descriptive name to help identify the purpose of this project

My Simics Project 1

Description

A brief description of this project and its purpose

Project description

Path

The path on the filesystem where project should be located

C:\source\github\ispm\logs\e2e-88681ca7\projects.e2e.ts-Project-support-Ir Browse

Simics Packages

Base

What Simics base package to associate project with.

1000-6.0.0 - Simics-Base

Simics-Base

Create Project

Figure 6 - Illustration of project creation page in My Projects

Press **Create Project** in the right corner when you are done.

The project detail view

When there are one or more projects, they can be selected in the sidebar list. Selecting a project will open the project details view which is used for managing and updating the selected project.

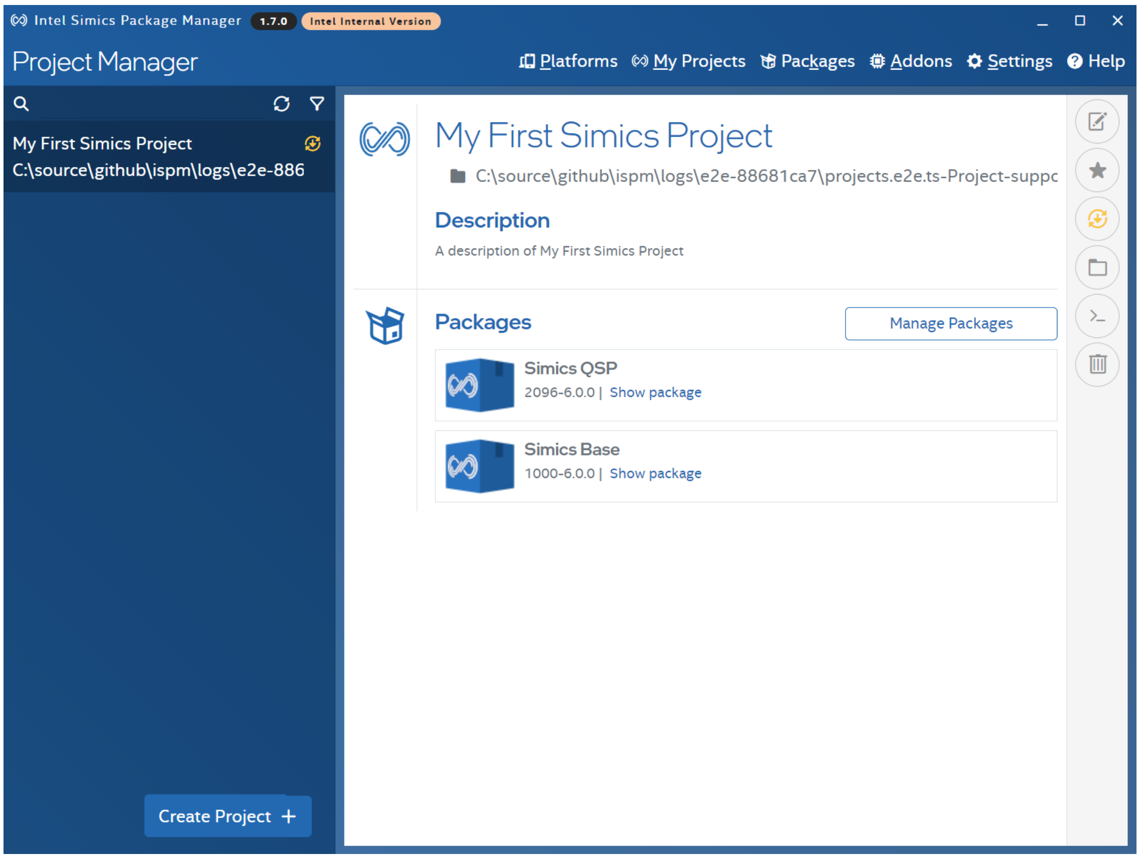

Intel Simics Package Manager 1.7.0 Intel Internal Version
Project Manager
Platforms
My Projects
Packages
Addons
Settings
Help
My First Simics Project
C:\source\github\ispm\logs\e2e-886
Create Project +
My First Simics Project
C:\source\github\ispm\logs\e2e-88681ca7\projects.e2e.ts-Project-suppc
Description
A description of My First Simics Project
Packages
Manage Packages
Simics QSP
2096-6.0.0 | Show package
Simics Base
1000-6.0.0 | Show package

Figure 7 - Illustration of project details page in My Projects

To the right of this view you have a few options:

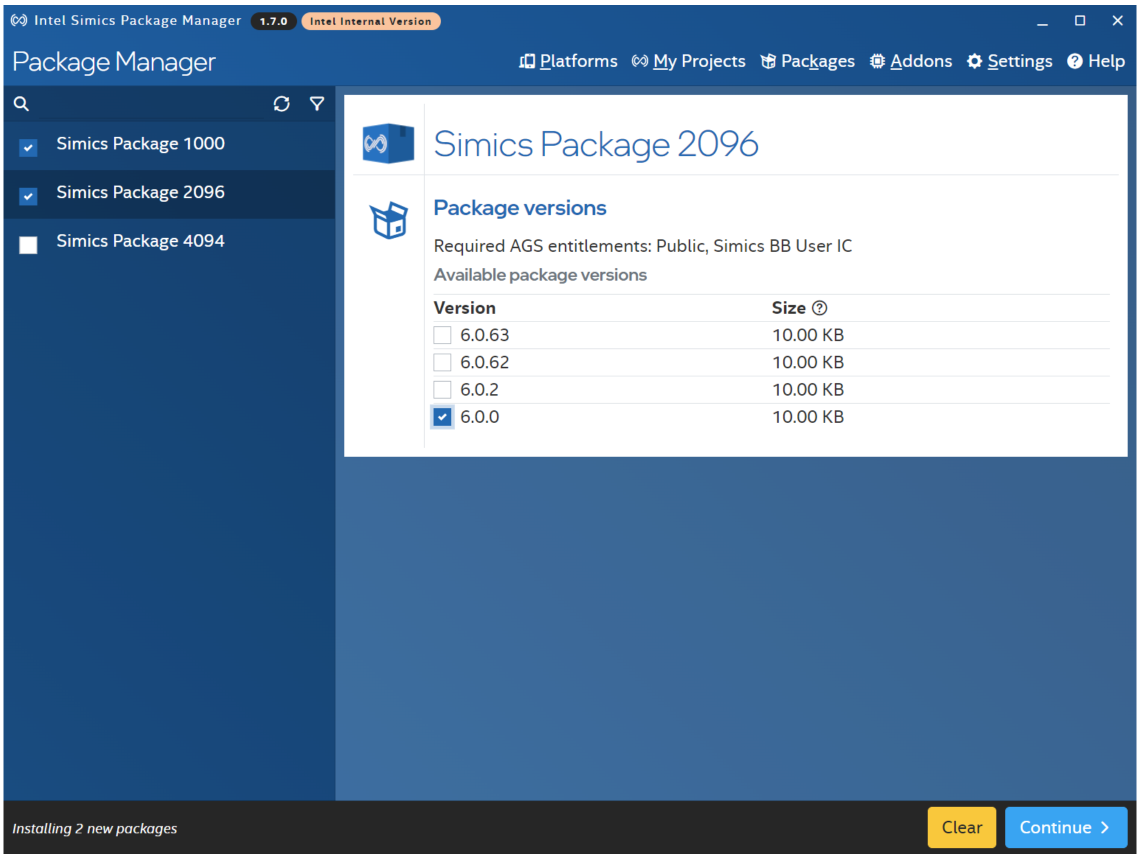
1. Edit project.
2. Mark as favorite.
3. Updates available.
4. Open in OS filesystem.
5. Open in OS specified terminal
6. Open in Eclipse* IDE for Intel Simics software
7. Delete Project.

Package Manager View

The package manager view is where you interact with packages. Initially the view will fetch all available package from your package repositories and install paths and then present them in a neat list.

Installation status bar

When you select a package version for either installing or uninstalling, you will get a status bar at the bottom of the package manager view. From this status bar you can then choose to clear all package versions checked for either installation or uninstallation or continue to the next step of the installation process.



Simics Package 2096

Package versions

Required AGS entitlements: Public, Simics BB User IC

Available package versions

Version	Size
☐ 6.0.63	10.00 KB
☐ 6.0.62	10.00 KB
☐ 6.0.2	10.00 KB
✓ 6.0.0	10.00 KB

Installing 2 new packages

Figure 9 - Illustration of the installation status bar in the bottom of the package view

Package details view

When selecting a package from the sidebar list, a detailed view of the package will be shown to the right. This view will look a bit different depending if the package has external metadata or if it is installed. The detail view contains:

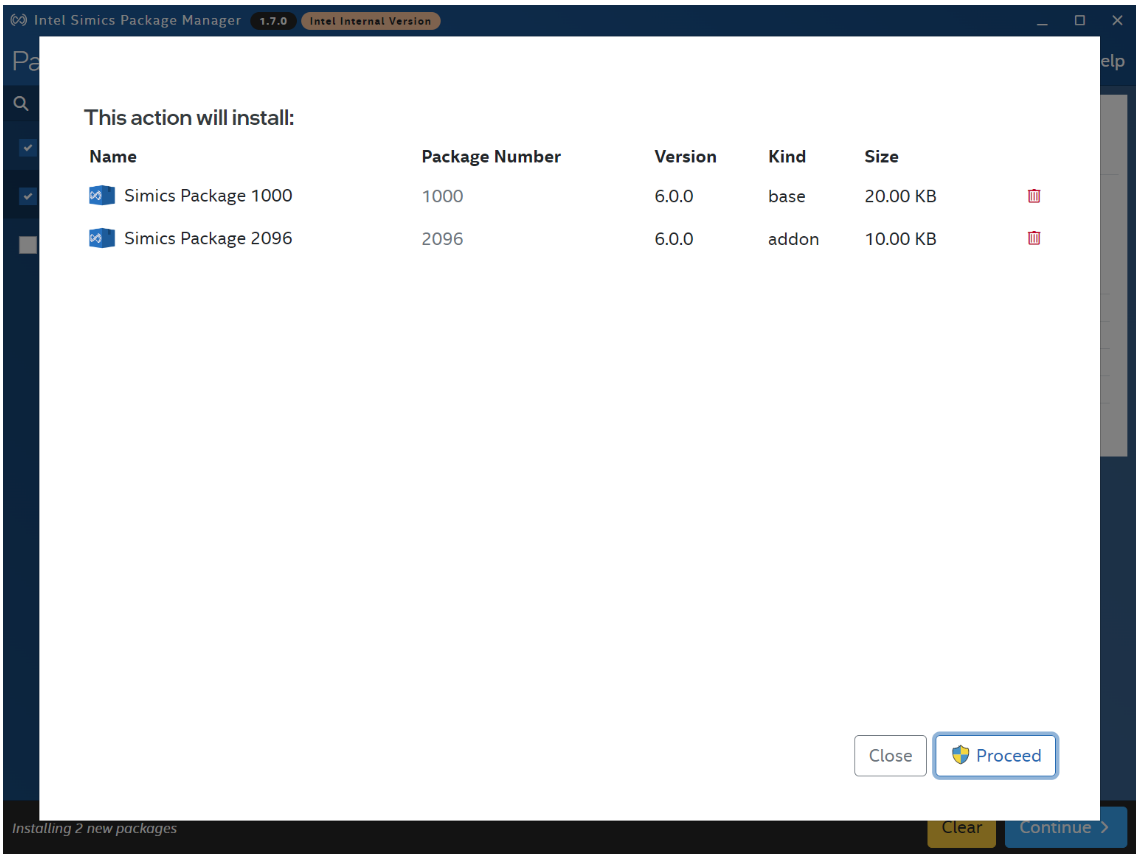
1. Package name, if package name is not available a package number will be shown.
2. Package description, will not be present if not available.
3. Status of what is included in package. Indicates if binary, source code and documentation is included. The info icon indicates that release notes are browseable for specific package version.

Installation confirmation view

You will be greeted with the following view when you have decided on what packages you want to install or uninstall and then pressed the **continue** button in the installation status bar.

This will give you an overview of the packages and their versions, as well as what actions you have requested. Last minute changes can be done by removing the action you do not want performed by clicking the respective trash can.

This is the same Confirmation View that you will be prompted with if you choose to install a Platform Manifest, Package Bundle or Package Repository.



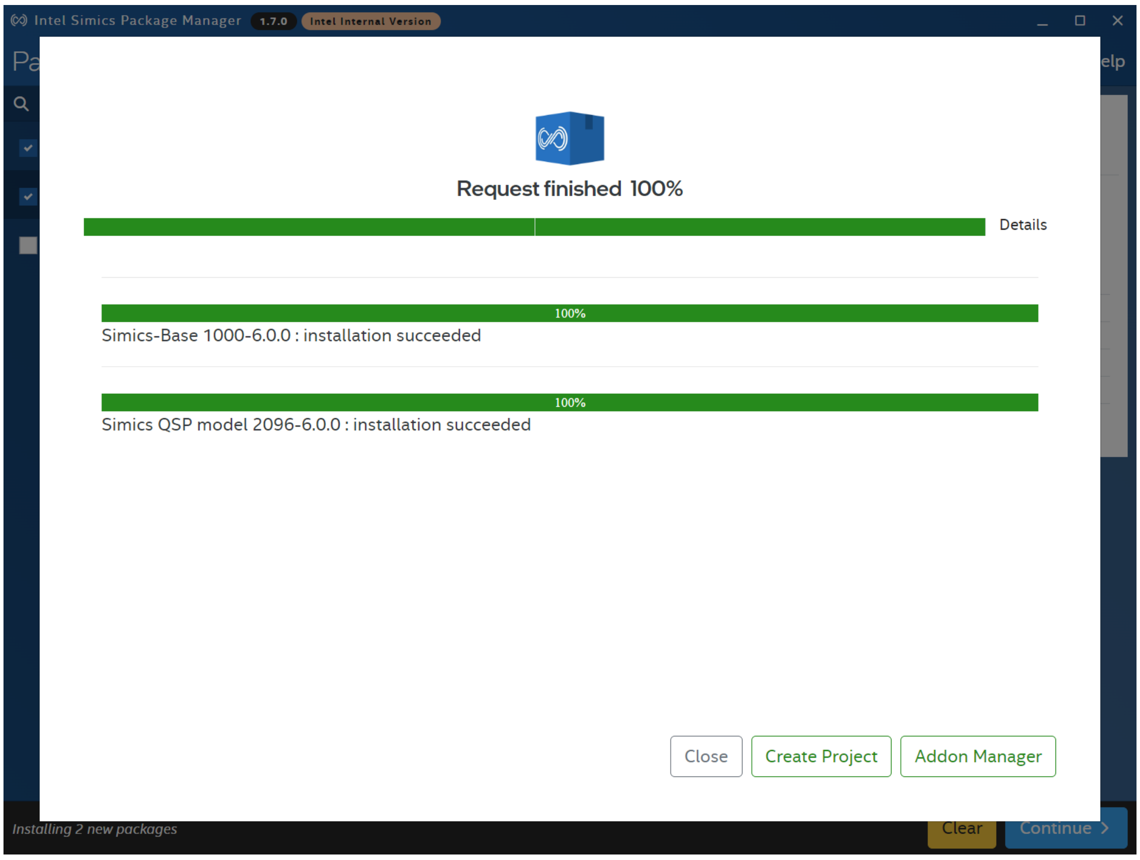
This action will install:

Name	Package Number	Version	Kind	Size
Simics Package 1000	1000	6.0.0	base	20.00 KB
Simics Package 2096	2096	6.0.0	addon	10.00 KB

Figure 10 - Illustration of the installation overview

To start installing an uninstalling press the **Proceed button** in the right corner.

To view details during installation press the **details button** to the right of the progressbar.



Request finished 100%

Simics-Base 1000-6.0.0 : installation succeeded

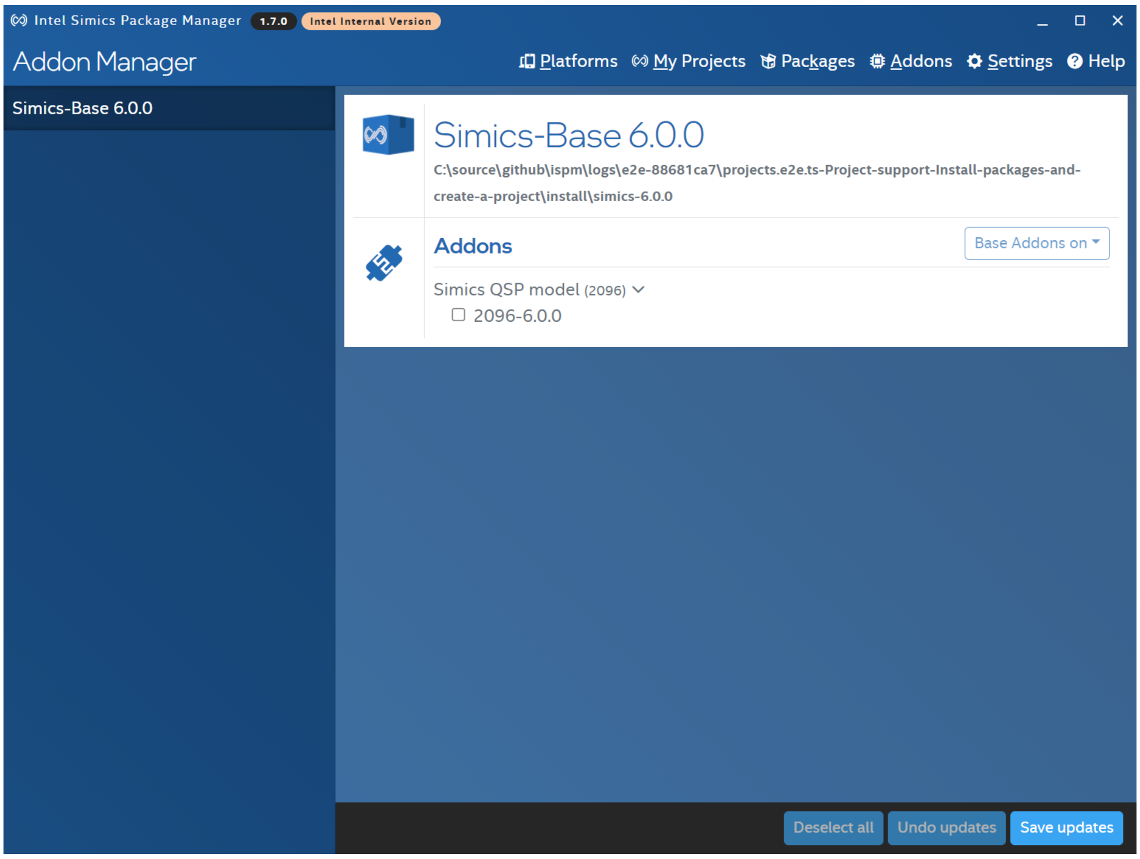
Simics QSP model 2096-6.0.0 : installation succeeded

Figure 11 - Illustration of the installation overview details

Addon view

We have added the addon manager to the new package manager in order to to remain compatible with old project creation flows both in Eclipse* IDE for Intel Simics software and standalone installations of the simulator software.

Note that creating a project through the installer does not require the use of the addon manager since the package list files are generated at the project level. Like previously, a base package is required for the addon manager to work. You can only bind one version of each addon package number to a base package. The packages also requires that your base and addons are of the same major version. It is also possible to copy addons from another base package and it will be indicated under the name if available.



Simics-Base 6.0.0

Simics-Base 6.0.0
C:\source\github\ispm\logs\e2e-88681ca7\projects.e2e.ts-Project-support-Install-packages-and-create-a-project\install\simics-6.0.0

Addons

Simics QSP model (2096)
☐ 2096-6.0.0

Figure 12 - Illustration of the addon manager

Settings

The setting page contains two different sections:

General settings

The general view is the default view when you navigate to the settings page. In this view you can edit package repositories and installation paths.

Package repositories

1. To change the priority of package repositories use the arrows in front of each repository path.
2. The Action buttons contains Edit path, Enable/Disable and Remove.
3. To input a new repository use the "Add repository path field" in the bottom of the Package repository section.

Platforms Repositories

Similar to Package repositories, the repositories that contain published `.smf` manifest files can be specified here in the same manner.

Installation Directory

Change your installation directory or remove it. Note, you need to have an installation directory active set in order to be able to download packages.

Default Project Path

This section lets you change what default path will be selected when you create projects. The default path is currently Your Home Directory + Simics/Projects. The Reset Button restores this default path if you have made changes to this setting.

Read only installation paths

Read only installation paths are locations with installed packages, typically installed out of your control, such as by a network administrator or installed in a shared location. By design, you are not permitted to uninstall packages found in read only installation paths.

Local Package Keys

Intel Simics packages may use encryption to protect sensitive information, and must be decrypted with a package key. You must either provide a keyfile containing the keys mapped to the versions of packages you wish to install, or the installer will prompt you for a decryption key for each attempt to install.

This file needs to be formatted with lines matching any of the following formats:

```
<packageNumber> <packageKey>
# e.g. 1000 ABC123...
<packageNumber>-<packageVersion> <packageKey>
# e.g. 1000-6.0.50 ABC123...
<packageName>-<packageNumber>-<packageVersion>-<host>.tar.gz.aes <packageKey>
# e.g. simics-pkg-1000-6.0.50-win64.tar.gz.aes ABC123...
<packageName>-<packageNumber>-<packageVersion>-<host>.msi <packageKey>
# e.g. simics-pkg-1000-6.0.50-win64.msi ABC123...
```

A malformed key file will result in an Invalid Key File error.

Advanced Settings

Multi User Configuration

When using a single package 1001 (Eclipse* IDE for Intel Simics software) installation between multiple users, configuration settings can clash. Using this setting *before* installing package 1001 in this configuration makes the configuration

directory read only and Eclipse* IDE for Intel Simics software then falls back to using per use configuration settings.

> This setting only applies to packages installed after the setting has changed.

Cache

When the package manager fetches package information from servers, it stores the information in a local cache to not overload servers and to decrease latency when performing operations in the installer.

From within the package manager it is possible to clear the current cache and to modify the length of time the cache is valid (default is 1 hour or 3600 seconds).

Proxy Settings

The proxy settings can be configured here to allow the installer to communicate either without any proxy, using your environment settings, or by manually overriding the configuration within the installer.

Path to Powershell (Windows only)

For Windows users, the application makes use of Powershell to determine whether the application has certain privileges.

If your path to Powershell is different than the default, you can adjust that path here.

Temporary Directory

As part of normal operation, the package manager must download and store files locally in certain use cases, and users have the ability to customize to suit their own infrastructure and use case.

Users have the ability to restore this setting to the app default if it has been modified.

Help

Third Party Licenses

This view shows what third party dependencies are used and their licenses.

Limitations

The view contains information about the what limitations there are in the current version.

Release notes

Contains release note from the current release of the package manager.

The Command Line Interface

Warning: Using the new installer on Windows can modif y existing packages on the f ile system. Be cautious not to change packages if you are not sure about what you are doing. Changes might be irreversible.

The command line interf ace of the package manager is its own separate binary

For help regarding run:

```
ispm -h
```

Which will provide you with the f ollowing:

```
Usage: ispm.exe command [global and/or command options]


Intel® Simics® Package Manager v1.9.1 (ispm.exe) is a command line
application for retrieving, downloading, installing, and managing Simics
Packages, Platforms and Projects.

This is an Intel internal release only and is not intended for use outside
of Intel's network.

For support, please contact simics-installer@intel.com or visit goto/ispm.

Options:
  -V, --version                       Display version information (default: false)
  --verbose [logLevel]                print all log output to console, optionally
  --logFile <filename>                set path to log file
  -h, --help                          display help for command

Commands:
  packages [options]                  install, uninstall and view information abou
  projects [options] [project-path]   create, edit and remove projects
  platforms [options]                 view, install or create projects based on av
  config [options]                    [Deprecated] view and edit various settings
  settings [options]                  view and edit various settings
  install [options] <pkgs...>         install packages directly
  uninstall [options] <pkgs...>       uninstall packages directly
  update-check [options]              check for updates to Intel Simics Package Ma
  about [options]                     display important information about this app
  help [command]                      display help for command

Read More Details for commands:
  You can get more in depth details of each command by running

  ispm.exe <command> --help

Common Examples:
  Install latest 6 Base package using install directory and package-repo settings
    ispm.exe install 1000-6.latest

  Non-interactively install all packages from --package-repo directory to --insta
    ispm.exe packages --install-all --package-repo /nfs/location/directory --inst

  Create a new project in /my/local/folder folder using Base version 6.0.177 and
```

```
packages - if not already installed. Use install directory and package-repo set
  ispm.exe projects /my/local/folder --create 1000-6.0.177 1031-6.0.latest

Install all the packages corresponding to the latest release of the manifest gr
the --install-dir directory. Use package-repo settings from the config file.
  ispm.exe platforms --install qsp-x86-7 --install-dir /my/local/folder

Non-interactively uninstall all packages from --install-dir directory
  ispm.exe packages --uninstall-all --install-dir /my/installation/directory --
```

Examples

We have provided a few examples on how to use the package manager to help you get started.

Getting started

There are two paths to get up and running, installing packages from the packages tab, or by installing packages from a platform.

First, using the Packages tab:

Up and running with Packages

In this example we are going to:

- Setup a package repository path.
- Install Eclipse* IDE for Intel Simics software.
- Install a base package.
- Install an addon package.
- Create a new project.
- Open the created project in Eclipse* IDE for Intel Simics software.
- Run created project from a terminal.

First things first, if you have not setup an installation directory go ahead and do that either by providing it during the preparation stage or by changing it under **Settings -> General**.

Make sure you have the correct path to the package repository with the packages you wish to download when you set your installation path. In this example we are using one of the Internal Artifactory* repositories for packages. To edit, change, delete or add a package repository go to **Settings -> General**.

Now when you have added the package repositories that you need, it is time to download our first package. Go to **Packages**. Wait while packages are being fetched from you package repositories. Press the quick selection checkbox in the sidebar of the package named `Simics Package 1001`, The status bar will pop up showing installing 1 new package. Press **Continue** when ready.

After pressing continue you will be prompted with details of what will be installed, press **Proceed** to continue. (On Windows you might be asked to run as administrator and get prompted with a UAC).

When your installation has finished **Close** the prompt.

Let's install the latest `Simics package 1000`, `Simics package 2096` and `Simics package 4094`

Follow the same steps as previously but this time when all packages have been installed press the **Create Project button** in the bottom right of the prompt.

This will take you to the create project page within **My Projects**. Give your new project a name.

Do not forget to add package 1001 addon at the bottom.

Press **Create Project** in the bottom right corner.

Your project should now have been created and you can view the project details.

Let's open your newly created project in Eclipse* IDE for Intel Simics software that you downloaded previously. The project detail page will detect if package 1001

is installed and will then enable the "open in Eclipse* IDE for Intel Simics software" button.

Press it and your project will open in Eclipse* IDE for Intel Simics software.

Congratulations! You now have a working project running in Eclipse* IDE for Intel Simics software!

Running the project from command line

If you want to run the simulator from the command line press the **open in terminal button** and your preferred terminal application will open.

Installing from a Virtual Platform

Next, we'll look at getting up and running by installing from a platform

So, you have received a Platform Manifest and now want to setup a project based on that manifest.

Navigate to the **Platforms view** and make sure your are in the **Install from manifest** section. *This should be the defalut section when you enter the Platforms view.*

Press the **Add new manifest from directory button** located in the manifest sidebar.

Select your manifest from your file system and press **Import Manifest**

When you import your manifest, it will automatically be selected.

Before you can go to the step to Create Project, you will have to download & install the packages described in the manifest.

Click on **Install Packages.

When packages are finished and installed press Create Project. Create your project as in the previous example and run the project as you like.

Batch Mode and the Command Line Interface

Taken from the examples given when running

```
ispm.exe -h
```

Command Line Interface Examples

Adding the CLI to your path

Currently the installer for the app does not add the CLI executable to your system PATH by default. If you would like to add it yourself these are the steps needed:

Add to PATH on linux (Bash shell)

Open your `.bash_profile` `.profile` or `.bashrc` (depending on your distribution) file with which text editor you prefer in this example we will use nano

```
nano ~/.bash_profile
```

Then add the path for the CLI to your path variable

```
export PATH="/path/to/ispm-directory/:$PATH"
```

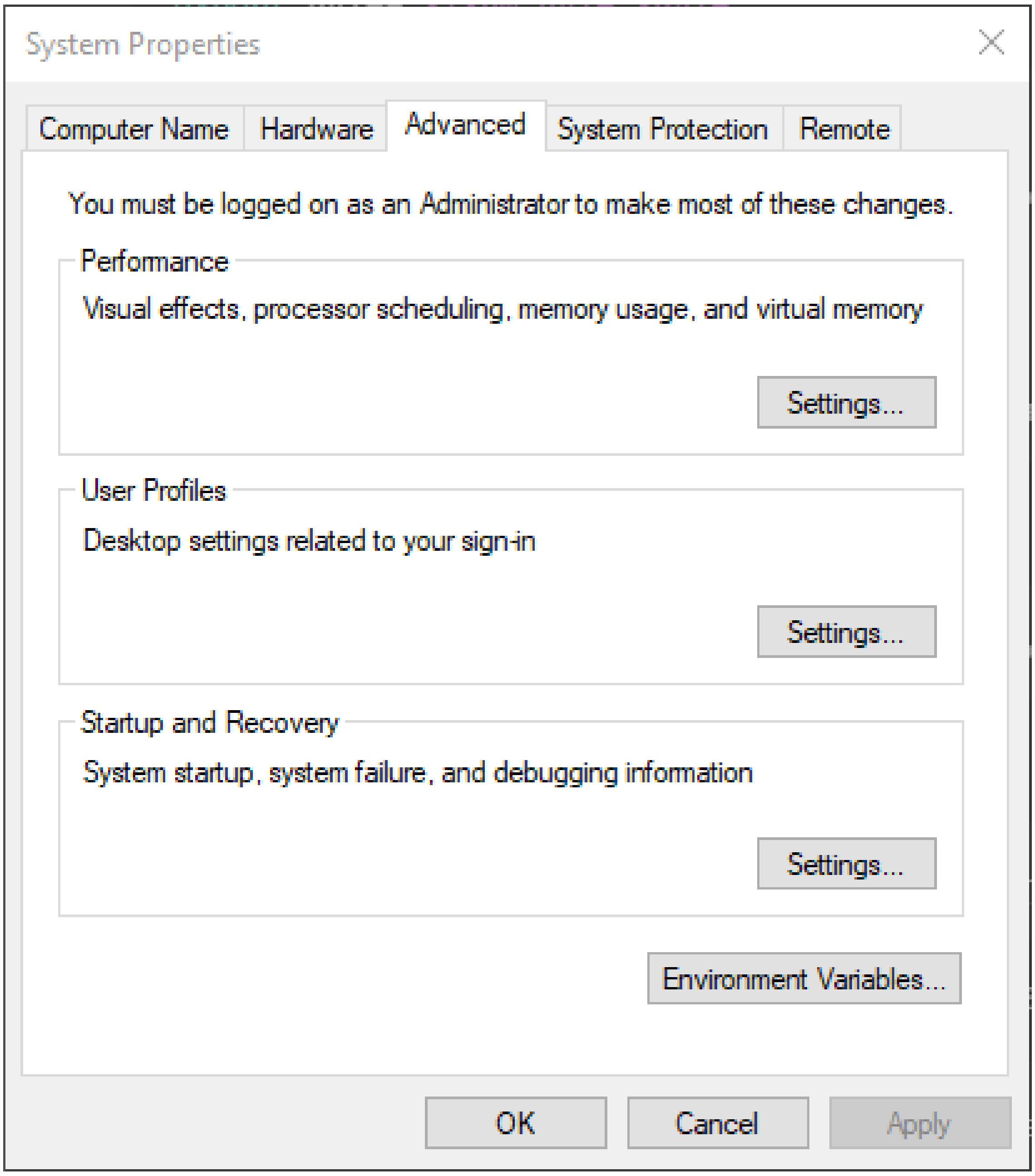
an alternative is to store the path as its own variable. This will make it a lot easier to read and change at a later stage.

```
export ISPM_PATH="/path/to/ispm-directory"
export PATH="$ISPM_PATH:$PATH"
```

Add to PATH on Windows

Open the `start` menu and type `system environment` and press the `Edit the system environment vaiables` result. This will open the system properties.

Press the button labeled `Environment Variables...`



You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

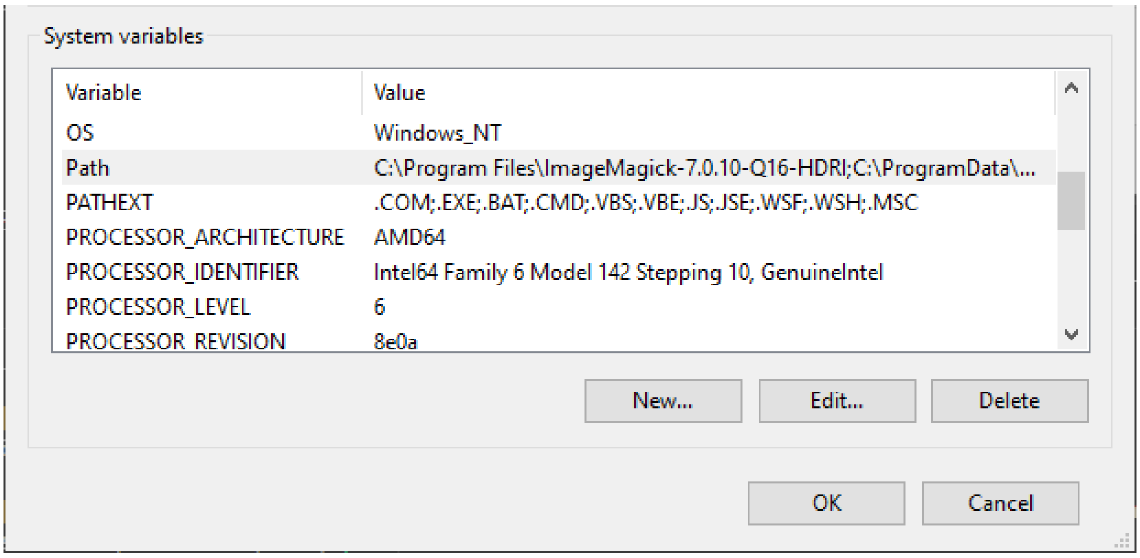
Desktop settings related to your sign-in

Startup and Recovery

System startup, system failure, and debugging information

Figure 14 - Image of system properties

Highlight the environment variable called Path for either the user or the system and press the e dit button

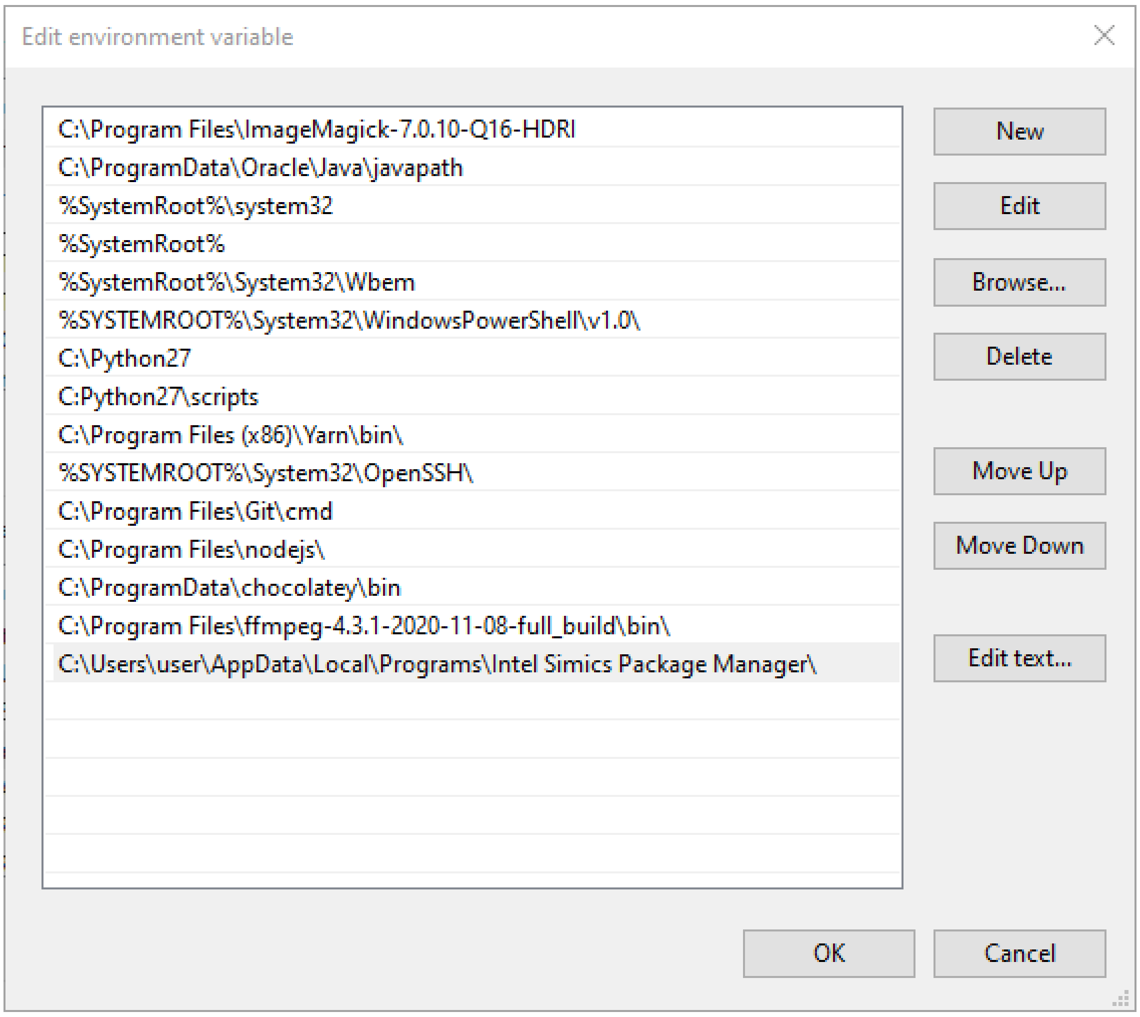


System variables

Variable	Value
OS	Windows_NT
Path	C:\Program Files\ImageMagick-7.0.10-Q16-HDRI;C:\ProgramData\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 10, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	8e0a

Figure 15 - Image of system environments

Then press new and add the path to the app path on a new line.



Edit environment variable

```
C:\Program Files\ImageMagick-7.0.10-Q16-HDRI
C:\ProgramData\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Python27
C:Python27\scripts
C:\Program Files (x86)\Yarn\bin\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Program Files\nodejs\
C:\ProgramData\chocolatey\bin
C:\Program Files\ffmpeg-4.3.1-2020-11-08-full_build\bin\
C:\Users\user\AppData\Local\Programs\Intel Simics Package Manager\
```

Figure 16 - Editet Path environment on windows

If you installed it for just your user it is likely located in `C:\Users\[me]\AppData\Local\Programs\Intel Simics Package Manager` and if you installed the app for all users, the path will instead likely be `C:\Program Files\Intel Simics Package Manager`.

Now you are ready to use the CLI version of the app from any terminal.

> You may need to restart existing terminals to properly get the new path value.

Make sure everything works by testing this in the terminal of your choice:

```
ispm.exe --help
```

Configuring the CLI

The CLI version of the app shares the same configurations as the GUI version of the application. This is to make sure that they stay consistent between uses. You are able to change/view this setting from the CLI version.

To View and list all settings and their values use this command:

Linux:

```
ispm config
```

Windows:

```
ispm.exe config
```

To get an overview of what options are available for the `config` command you can use `--help`

Linux:

```
ispm config --help
```

Windows:

```
ispm.exe config --help
```

The ´config´ command has subcommands for each setting that is possible to edit. These are the subcommands available:

- install-dir [path]
- package-repos [--add, --remove, --enable, --disable] [path]
- ro-install-paths [--add, --remove, --enable, --disable] [path]
- proxy [--https, --no-proxy, --dont-use, --use-env]

By omitting the flags of a subcommand, the cli will list the current configuration set for that setting.

install-dir

The `install-dir` subcommand specifies where your Packages will be installed to. If you have used the GUI version you will be given a suggestion for installation path. This is not the case for the CLI version. We suggest that you add this path as your installation path The first time you start the CLI version:

Linux:

```
ispm config install-dir /where-ever-you-see-fit
```

Windows:

```
ispm.exe config install-dir C:\Users\[you]\AppData\Local\Programs\Simics
```

package-repos

If you are primarily installing packages from network or local disk based package repositories, you can use the ´package-repos´ subcommand to enable, disable, add and remove repositories. For more info about Package repositories see Chapter `Package Repository`.

To **add** a package repository:

Linux:

```
ispm config package-repos --add [url or path to repository]
```

Windows:

```
ispm.exe config package-repos --add [url or path to repository]
```

To **remove** a package repository:

Linux:

```
ispm config package-repos --remove [url or path to repository]
```

Windows:

```
ispm.exe config package-repos --remove [url or path to repository]
```

To **enable** a package repository:

Linux:

```
ispm config package-repos --enable [url or path to repository]
```

Windows:

```
ispm.exe config package-repos --enable [url or path to repository]
```

To **disable** a package repository:

Linux:

```
ispm config package-repos --disable [url or path to repository]
```

Windows:

```
ispm.exe config package-repos --disable [url or path to repository]
```

ro-install-paths

If an administrator has prepared and installed packages in a common location for you to use in your projects, you can manage those read only installation paths with the `ro-install-paths` subcommand.

The `ro-install-paths` uses the same flags as the `package-repos` subcommand.

To **add** a readonly installation path:

Linux:

```
ispm config ro-install-paths --add [path to installed packages]
```

Windows:

```
ispm.exe config ro-install-paths --add [path to installed packages]
```

To **remove** a readonly installation path:

Linux:

```
ispm config ro-install-paths --remove [path to installed packages]
```

Windows:

```
ispm.exe config ro-install-paths --remove [path to installed packages]
```

To **enable** a readonly installation path:

Linux:

```
ispm config ro-install-paths --enable [path to installed packages]
```

Windows:

```
ispm.exe config ro-install-paths --enable [path to installed packages]
```

To **disable** a readonly installation path:

Linux:

```
ispm config ro-install-paths --disable [path to installed packages]
```

Windows:

```
ispm.exe config ro-install-paths --disable [path to installed packages]
```

decryption-key-files

Certain packages require decryption keys to install. You may specify a local file containing these decryption keys with the `decryption-key-files` subcommand. If this file is not specified, you may be asked to manually enter decryption keys during the installation.

The `decryption-key-files` subcommand also shares the same flags as previous subcommands.

To **add** a decryption key file:

Linux:

```
ispm config decryption-key-files --add [path to key file]
```

Windows:

```
ispm.exe config decryption-key-files --add [path to key file]
```

To **remove** a decryption key file:

Linux:

```
ispm config decryption-key-files --remove [path to key file]
```

Windows:

```
ispm.exe config decryption-key-files --remove [path to key file]
```

To **enable** a decryption key file:

Linux:

```
ispm config decryption-key-files --enable [path to key file]
```

Windows:

```
ispm.exe config decryption-key-files --enable [path to key file]
```

To **disable** a decryption key file:

Linux:

```
ispm config decryption-key-files --disable [path to key file]
```

Windows:

```
ispm.exe config decryption-key-files --disable [path to key file]
```

proxy

If you are behind a proxy, then you can configure the package manager with the `proxy` subcommand to allow it to access packages or platforms outside the proxy.

The proxy subcommand have four flags:

- `--https` which is used for setting the proxy url to route the network through
- `--no-proxy` specifies the comma separated host patterns for which to ignore proxy settings. e.g..
- `--dont-use` Use no proxy at all
- `--use-env` Use the current evironment HTTP_PROXY, HTTPS_PROXY and NO_PROXY

Installing through the CLI

Installing a Package Bundle

To install a bundle it is important that you have setup an installation Dirctory otherwise this command will fail.

If you have not setup an Installation Directory you can use the flag `--install-dir` to temporarly set one up.

Linux:

```
ispm packages --install-bundle /path/to/bundle.ispm
```

Windows:

```
ispm.exe packages --install-bundle c:\path\to\bundle.ispm
```

> Note: The bundle flag can currently only access bundles local to your file system.

https paths will not be accepted.

Creating a project with your bundle

To Create a project based on your bundle you can use the `--create-project`.

Linux:

```
ispm packages --install-bundle /path/to/bundle.ispm \
--create-project /path/to/project
```

Windows:

```
ispm.exe packages --install-bundle c:\path\to\bundle.ispm ^
--create-project c:\path\to\project
```

Install latest base package

This example is using the package repositories and installation directory you have setup in your config.

Linux:

```
ispm packages --install 1000-6.latest
```

Windows:

```
ispm.exe packages --install 1000-6.latest
```

Installing a specific base package version

This example is using the package repositories and installation directory you have setup in your config.

Linux:

```
ispm packages --install 1000-6.0.30
```

Windows:

```
ispm.exe packages --install 1000-6.0.30
```

Uninstalling a base package

This example is using the installation directory you have setup in your config.

Linux:

```
ispm.exe packages --uninstall 1000-6.0.24
```

Windows:

```
ispm.exe packages --uninstall 1000-6.0.24
```

Combining uninstalling and installing a base package

This example is using the package repositories and installation directory you have setup in your config.

Linux:

```
ispm packages --install 1000-6.0.30 --uninstall 1000-6.0.24
```

Windows:

```
ispm.exe packages --install 1000-6.0.30 --uninstall 1000-6.0.24
```

Installing/uninstalling all packages from a repository

It is possible to install all packages from a package repository by using the `--install-all` flag in combination with the `--package-repo` flag

Linux:

```
ispm packages --install-all --package-repo /nfs/location/directory
```

Windows:

```
ispm.exe packages --install-all --package-repo /nfs/location/directory
```

It is also possible to uninstall all packages with `--uninstall-all` from either your install directory specified in your config or by using the `--install-dir` flag to specify specific location.

Linux:

```
ispm packages --uninstall-all --install-dir /my/installation/directory
```

Windows:

```
ispm.exe packages --uninstall-all --install-dir /my/installation/directory
```

Non-interactively

You can specif y that you want to run your commands Non-interactively by using the `--non-interactive` f lag with your commands.

Linux:

```
ispm packages --uninstall-all
--install-dir /my/installation/directory
--non-interactive
```

Windows:

```
ispm.exe packages --uninstall-all
--install-dir /my/installation/directory
--non-interactive
```

*****_Intel and Simics are trademarks of Intel Corporation or its subsidiaries._ *****_Other names and brands may be claimed as the property of others._