Remote Software Toolkit Installer

Software Management for the ReST of us.

Eric T Meek, Innovative Computing Laboratory, UT
`<meek@cs.utk.edu>`
Jeff M Larkin, Innovative Computing Laboratory, UT
`<larkin@cs.utk.edu>`
Jack Dongarra, Innovative Computing Laboratory, UT
`<dongarra@cs.utk.edu>`

Revision History
Revision $Revision: 1.8 $ $Date: 2005/06/06 15:42:00 $

Abstract

Consumer software installation "Wizards" have become the de-facto standard for installing software in a workstation (homogeneous) environment. This capability, however, has been absent from most scientific and distributed software. This technical report introduces the ReST Installer as a flexible installation wizard for both source and binary distributed applications that have not previously been able to provide a simple installation mechanism.

Table of Contents

Project Introduction

As grid computing has become a widespread area of research, the groundwork has been laid for many new and thought provoking ideas. However as with most new ideas and technologies, certain aspects of grid computing have been overlooked causing the usefulness to seemingly lag far behind the potential. One such aspect in grid computing is that of software distribution. In grid software, the idea of using computing resources that are geographically separated and architecturally diverse is simple, however, the logistics of installing software across in such an environment can be both tedious and frustrating. Clearly there is a need for a simple mechanism handling the installation and maintenance of software in such environments.

The Remote Software Toolkit (ReST) was originally conceived as a solution to installing, maintaining, and monitoring large grid software environments. Since grid computers often consist of a wide variety of machine architectures, it is inherent that scientific software be distributed in a form supporting as many architectures as possible. Facilitating such broad architecture support, however, generally requires software be distributed as source based packages. Distributing in this manner allows the software to be tuned to each specific architecture but also greatly increases the complexity of the installation process. Although the Remote Software Toolkit (ReST) was initially conceived for grid environments, as ReST matured it became clear that they could be useful well beyond just grid computing. ReST has become a robust way of supporting any software, not just grid software, across large collections of machines.

Software installation in a heterogeneous environment generally requires a user to login to numerous machines and perform a similar process on each. Making things worse, source-distributed software installation often requires a lengthy and confusing configuration, compilation, and installation process. The ReST Installer provides a familiar, wizard-like interface for source and binary distributed software that can be used to install software easily and automatically, even in a large heterogeneous environment. At the core of the ReST Installer is the ReST package model, described in more detail later in this document. Allowing software describing their software in ReST software packages, software maintainers can easily define their software's requirements, options, and configuration process as an "Installation Wizard" using the ReST Installer. This wizard presents users the software's complex options in a simplified manner, shifting the burden of software installation from the user to the software provider.

The ReST Framework

In order for the ReST Installer to work in virtually any environment, several logistical challenges had to be addressed. For heterogeneous environments, three major logistical challenges existed in software distribution. First, a common interface for connecting to the remote machines had to be selected. Second, a sane installation environment needed to be created and maintained on the remote machines. Finally, a package model that could describe diverse software packages and their requirements was needed. Graphical challenges also existed, specifically the goal of providing a simple installation wizard for complex, heterogeneous environments proved to be a difficult undertaking. Addressing these challenges resulted in the creation of the ReST Framework, the basis for the ReST Installer and other components of the Remote Software Toolkit.

Communication

When selecting a communication mechanism for ReST, it was crucial to select a service that is secure, widely available and simple to use. Choosing a communications scheme widely available was important for two reasons. First, because most system administrators are hesitant to introduce new services to their machines and second choosing a widely available service will help drive the adoption of ReST. For these reasons the secure shell (SSH) protocol was selected as the default communications framework of the ReST Installer and future ReST applications. Selecting SSH allowed the leveraging of existing software libraries both accelerating the development of the ReST Installer while providing system administrators piece of mind about security. Although SSH is arguably the most widely used communication mechanism, it may not be available for all environments. With this is mind ReST was designed to accept communication plugins to support any desired communications framework.

Packages

As development on the ReST Installer began, it was quickly determined that no packaging systems existed completely meeting the requirements of ReST. Existing systems primarily focused on either source or binary software distribution in a homogeneous environment with none providing a mechanism for parameter customizing that would appear in the graphical wizard. Since an installation may be on a heterogeneous set of machines, the packaging scheme also needed to handle the possibility that the software will be installed on a wide range of machines simultaneously. With no packaging systems meeting these needs, a new package specification was designed allowing maximum flexibility to package developers while maintaining ease of use. The ReST package specification gives developers a way to completely describe the installation process for their software, including configuration options, as either source or binary packages. Often times the software user is not fully aware of options or common problems that may occur or be available to them during installation. The ReST package specification encourages developers, who have an intimate understanding of the software, to package their software in a manner fully allowing the user to configure and install the software with minimal effort. Developers may define their configuration options and configuration file formats in the package so that users may be presented the options in an understandable manner.

The most important part of a ReST package is the xml description file (package.xml). The package.xml file describes all aspects of the software package. The first aspect described is the software contained in the package. The second aspect described is the installation process and options available to the user. Finally, it describes action the user will be able to perform on the remote location once the package is installed. By encapsulating this information in an XML file, maximum flexibility is maintained while keeping machine-readability. The package.xml file is broken down into several sections: the package header, package actions, and installation and uninstallation "steps". Each of these sections is described below.

Header

As should be expected, the header section contains basic meta-data about the software package. Information such as the software name, version, web page, readme, and license is located in the package header. Additional information about the packager can also be placed in the package, for example, if it is packaged by someone other than the original author. The header also contain URIs to the actual software sources, patches, and configuration files, which are all usually included in the package. If configuration files are included, the header will contain the definitions required to edit the file contents. An example of a package header appears below.

Example 1. A ReST Package Header

```
<!-- Basic information about the software package -->
<header>
  <name>NetSolve</name>
  <version>2.0</version>
  <description>NetSolve is a grid middle-ware package</description>
  <uri>http://icl.cs.utk.edu/netsolve/</uri>

  <!-- Basic information about the packager -->
  <packager>
    <name>Jeff M.  Larkin</name>
    <uri>mailto:larkin@cs.utk.edu</uri>
  </packager>

  <!-- Package source(s).  We can do both remote and local files  -->
  <packagesrc>NetSolve-2.0.tgz</packagesrc>

  <!-- Configuration files that need editing -->
  <configfile packagefile="server_config"
              remotefile="NetSolve-2.0/server_config"
              description="NetSolve Server Configuration File">
    <sub name="agent" description="The NetSolve Agent hostname"
                default="netsolve.cs.utk.edu"/>
  </configfile>
</header>
```

Package Actions

Actions define what a software can do once it has been installed. Software packages such as libraries may have only a few, if any, actions, while software such as servers could have many. The ReST Installer allows users to select actions to be run on locations once the installation has completed, such as starting a server or running a test suite. Future ReST applications will allow users to run these actions after installation using a simple graphical browser. Actions are defined similarly to commands, but are run after the package has been completely installed on a location. An example actions section appears below.

Example 2. Package Actions

```
<actions>
  <action name="Start Server" tooltip="Start a NetSolve server.">
    <command value="/bin/bash ./start_server.sh"
    statusmsg="Starting Server" errormsg="Failed to start server."/>
  </action>
<actions>
```

The 6 Steps

The process of installing a package is described in five steps and uninstallation is described in one, optional step. Each of the first five steps are essentially the same, consisting of zero or more commands, each of which consists of zero or more options. The steps simply provide a way of logically grouping the commands. Here is a description of each of the first five steps in the order that they appear in the file (also the order that they will be executed). It should be noted that file transfer/substitution is not one of the six steps but takes place between preparation and configuration.

1. *Preparation* - Commands used to prepare for future steps, (i.e. decompressing source archives)
2. *Configuration* - Any pre-compilation steps. (i.e running a GNU configure script)
3. *Compilation* - Source packages should be compiled during this step. (i.e. make all)
4. *Installation* - Files are copied to their final location during this step. (i.e. make install)
5. *Completion* - Post-installation clean-up occurs during this final step. (i.e. removing source directories)
6. *Uninstallation* - Commands for uninstalling the package.

Many considerations must be made when installing the software on the remote systems. The most common, and perhaps most difficult, consideration is the existence of shared filesystems and a common architecture, like that found on computing clusters. On such machines certain operations may only need to be performed once and others may need to be performed on all machines. Likewise files may only need to be copied once to be used on all machines. For this reason commands in each step may be grouped so that contention will not occur.

Graphical User Interface

Because most scientific, grid, and open source software lack the friendly graphical installation process of most commercial software, users are forced to endure a complex and often confusing installation process. Thus, access to most scientific and grid software is limited to users with an in-depth knowledge of the installation package and the operating environment to be used. Most commercial and some free software packages address this problem through the use of installation wizards. Installation wizards, a common feature in homogeneous environments, are the basis for the design of the Remote Software (ReST) Installer. Reducing the complexity and depth of knowledge required to install even the most basic software package was a fundamental key. Accomplishing this required a simple yet extensible installer, able to handle the most basic to the most complex software packages with minimal effort on the part of the user and packager. Future ReST applications will follow this philosophy to provide a friendly interface for interacting with software that has already been installed.

Initially designing the ReST Installer mirroring the simplicity of an "install wizard" for a single, known (homogeneous) system proved to be a bit naive. Designing a simple installation interface for a heterogeneous environment proved to be a larger challenged than it first appeared. Supporting installs in a such an environment required graphically exposing all the intricacies of the package installation. This fact alone increased the complexity of the ReST Installer beyond the typical homogeneous environment's install experience of clicking on next until done. Exposing the package intricacies requires the users have specific knowledge of both the install package and locations, far exceeding any requirement in a homogeneous installer. This made providing a simple way of displaying the required information key to the success of the ReST Installer.

The complexity of exposing package intricacies has been lessened; however, through various graphical means in the ReST Installer. Source based package complexities generally fall under one of two areas, file substitutions and various command line options. The ReST Installer defines these two areas as advanced and are by default hidden from the user -- hiding the complexities. In order to facilitate hiding the advanced options, packagers can specify default values for each advanced elements. The default values are then used if the user does not specify values for the advanced options. Default values may not work in all cases, but do allow the developers a way to make suggestions to the users. If it is necessary for the packager to force configuration of the advanced options, descriptions can be specified for each element helping the user make appropriate decisions.

ReST Installer Basics - A Walk Through

Installer Introduction

Designed to emulate as closely as possible the experience of a homogeneous installer, the ReST installer first displays a welcome screen (Figure A.1) with information pertaining to the installation package. If the package is provided with extra information it is displayed before the license panel in a similar layout as the License panel (Figure A.2). As an optional feature, the ReST Installer supports package license formats which either first require the user accept the license before proceeding or just display the license without forcing the user to accept. Figure A.2 shows the license panel requiring the user to accept the license before the *Next* button is enabled.

Saving an Install Script

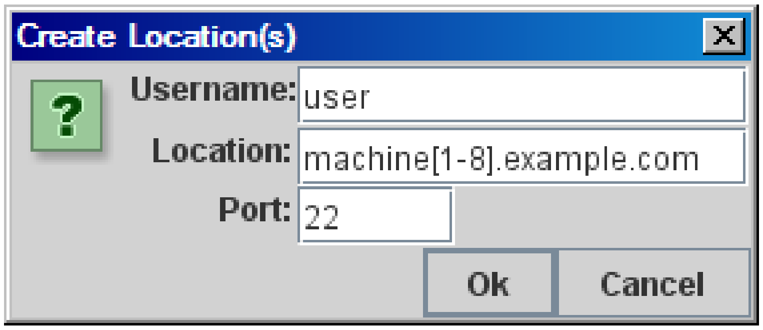
An extremely useful feature of the ReST Installer is the ability for users to create installation scripts duplicating an installation at some time in the future. This is useful for packages that update frequently, allowing the user to step through the installation without configuring groups or options. After an installation is complete, the ReST Installer will automatically ask if the current installation should be saved as an installation script upon closing the Installer. Once a script has been saved, future Installer sessions can make use of it by selecting it on the *Setup Locations* screen of the Installer and choosing *Import*.

It should be noted that it is possible for several groups to be configured using similar options and/or substitutions by configuring the first group as desired and then saving the install. Before the install is saved, all the previous panels should be completed as desired so the saved install is faithfully recreated. After saving the install, the generated script can be manually edited making duplication of groups options and substitutions as easy as a cut and a paste.

Setup Locations

Setup Locations (Figure A.3) is the first panel deviating from the traditional installer. Setup Locations allows for choosing one or more locations on which to install the package. This panel is designed to logically parallel the filesystem location selection panel or disk to install packages panel in a homogeneous installer. Unlike some homogeneous installations, the filesystem install location cannot be changed in the ReST Installer. In order to easily manage the installed packages and to create a sane working environment, ReST defines a location in the user's home directory (~/rest/) as the default location for building software. To successfully use the ReST Installer, each target location must have a valid account for the user and must be reachable via secure shell (SSH). Successful package installation, however, depends on the specific requirements defined by the package provider. Locations can be inserted using a regular expressions to define multiple locations with similar names as shown in Figure 1. Entering a location as someone@somewhere[0-2].example.com would create three locations of someone@somewhere0.example.com, someone@somewhere1.example.com, someone@somewhere2.example.com. If a group of machines will be used for the installation of several packages, the user may wish to write or save a script file containing the list of machines and options used during installation.

Figure 1. Create Locations



Username: user
Location: machine[1-8].example.com
Port: 22

The create locations dialog box filled in to create 8 different location user@machine1.example.com - user@machine8.example.com

Setup Logical Groups

Handling installations in a heterogeneous environment required the creation of logical groups (Figure A.4). A logical group consists of locations grouped by filesystems and/or binary compatibility. In other words, if five machines share a common filesystem and programs compiled on one machine may be run on all, then these machines should be placed in a logical group. By creating logical groups, the ReST Installer is able to reduce the network traffic, file space and time needed to install the package. Two important entities exist among logical groups: build masters and file masters. When installing a package, some parts of the package may only need to be executed on one machine for all machines in the group to benefit from that action. In this case the command will only be run on the build master of the group. Every group must have a build master. If no build master is explicitly selected by the user, one will be automatically chosen from the group. Once a build master is selected, a single asterisk will appear by its name. Explicitly choosing a build master is done by selecting one machine in the group and then pressing the *Set Buildmaster* button below the group list.

Multiple logical groups may share a common filesystem, making it possible to transfer files to one machine and have them available to all machines within multiple logical groups. For this reason the file master entities exist. A file master is an extension of a build master, so if a machine is a file master, it is also the build master of its logical group. A filemaster is created by choosing a machine and pressing the *Set Filemaster* button below the groups list. This will cause a window to appear, which will include all of the defined groups. To select multiple groups within this window, hold the *ctrl* key while clicking on the group name. Once all the the desired groups are selected, press *Ok*. Selecting a file master will only affect the build master of the group to which the new file master belongs. All other groups will remain unchanged. When a machine has been selected as a file master, two asterisks will appear by its name.

By default a logical group is created for all machines unselected in the groups panel. This group is known as RSI_DEFAULT and may be treated as a normal logical group or treated as separate machines, each in their own logical group. RSI_DEFAULT is treated as a logical group by default. To treat each machine as an independent group, simply uncheck the *RSI_DEFAULT grouped* checkbox on the groups panel and when the next button is clicked a new group will be created for each machine.

Setup Actions

Actions can be added to either a location or a logical group by using the Setup Actions panel (Figure A.5). Adding an action to a logical group will cause the action to be performed on each location in the group and is very useful in adding actions to many locations quickly. To add an action to a location or logical group, select the target of the action, click on the + button and then select the action(s). To remove actions, select the group or location with the actions, select the actions and then click on the - button.

Advanced Options

By default the ReST Installer runs in a simplified mode, which removes the need for users to configure specific package options. In basic mode packages are installed using defaults provided by the software packager. Often times, however, it is necessary to configure the software for the specific location on which it is being installed. If the advanced options are not enabled by default by the packager, the user can select *Advanced* from the *Options* menu enabling the File Substitutions and Configure Commands panels.

Setup File Substitutions

Some packages require certain configuration files to be edited before the package will run correctly or with special options enabled. The ReST Installer will display these files as a form that can be quickly edited (Figure A.6). Options that are either on or off are represented by a check box. The option may be a list of possible choices, which are represented by a drop-down box. Some options may require text input and are represented by either a single-line text box or a multiple-line text area. The process of editing the configuration files should be familiar to anyone who has filled-out a form on a web page. At this time file editing is done on a per-group basis and the current group can be selected from a drop-down list next to the *reset* button.

Configuring Command Options

Configuration of command options (Figure A.7) is done almost exactly like editing configuration files. It is worth noting that options requiring text input are represented by text boxes and that an empty text box signifies that the option is disabled. Enabling a text option requires simply clicking in the text box. If a default value is Provided by the packager, it will be inserted into the text box and highlighted allowing it to be replaced by just typing over it. Disabling a text option is done by simply removing any text from the text field. Just as with configuration files, command options are group-based and the current group selected from the drop-down list beside the *reset* button.

Setup Authentication

The final step before the software installation occurs is configuring how the ReST Installer will access the remote machines (Figure A.8). At this time ReST can only access machines via Secure Shell (SSH). The preferred authentication method is via public key authentication. Using SSH keys for authentication on the target locations is performed by pressing the *Add* button next to the "Private Keys" field. This will display a File Chooser window to select one or more OpenSSH formatted private key files to use for authentication. If SSH private keys have previously been used with the ReST Installer, the keys used previously will already be entered in the field. It is important to note that any key listed in the text field will not be removed when adding a new key. If a key is to be replaced, first it's path must be manually removed and then the new key must be added using the add button.

If key-based authentication is not an option, users may create password groups from the listed machines. A password group contains machines that require the same password for remote access. Creating password groups is done in the same way as creating logical groups; simply name the group for convenience then select the desired locations and press the >> button to move the locations into the new group. When the Installer tries to access the locations in this group, you will only be asked for the password once. Be warned, however, that this convenience is achieved by caching the password to memory, a potential security hazard. Users who do not want this behavior should use SSH keys for authentication.

The final step of configuring authentication is to point the ReST Installer to your current known_hosts file. When SSH makes a connection to an unknown machine it caches the server's key in a known hosts file to compare against at the next login. If you already have a known hosts file, use the *Select* button to choose the proper known hosts file. If however, a known hosts file is not available simply enter the name of the file that you would like to use. Using a known hosts file that has already been populated with keys avoids the dialog asking for authorization of the keys for the remote locations.

The Installation

Once the needed authentication credentials have been provided, a list of locations will appear with each target location's installation status. The icon to the left of the location name will show the machine status, either non started, in progress, warning, error, or completed. To the right of the location name appears in parenthesis what is currently being performed on the location or an error message. Once all the locations have been attempted and have either completed or failed, the next button will become available. Pressing the next button will display a final panel and change the next button to "Finish" causing the installer to exit when pressed.

If an error occurs during the installation process it may be useful to view the output generated on the location. ReST saves the output of the most recent installation to an XML file to provide feedback as to why failures occurred installing on specific locations. The output file is saved in the user's home area in the subdirectory *.rest/output/LOCATION.xml* , where *LOCATION* is the shortened hostname of the desired location. Windows users should know that the *.rest* directory will appear in *C:\Documents and Settings\USERNAME*. A web-based utility has been provided to give users an easy way to view this output file. This utility can be found at http://icl.cs.utk.edu/rest/under the *Tools* menu item.

Future Work

Several exciting areas of future research exist in the ReST Installer.

1. *Graphical Packager* - Currently, creating a ReST package requires developers to create an XML description of their software and then package it with the required files manually. A graphical packager would serve as a wizard for creating a ReST package walking developers through the process of creating ReST package.

2. *Local Installs* - The ReST Installer currently installs on local machines with a SSH daemon running; however, truly supporting local installs will require a new communications plugin.

3. *Intra-Package Dependencies* - In some cases selecting an installation option could cause one or more commands to need to be run or skipped. Likewise one command may require that another command already be completed. Future versions of the ReST Installer will support such dependencies.

4. *Inter-Package Dependancies* - Future versions of the ReST Installer will allow packages to depend upon other packages. Although this will cause many new complexities in the ReST back-end, it will make ReST more competitive with other package managers and easier to use for both users

and packagers.

5. *Reworking ReST Registry* - When installing on many locations, the ReST Registry can become long and slow to parse. Reworking the registry design, learning from current experience, could optimize it structure by moving information out of the it not accessed much and only leaving the most important.

6. *ReST environment* - ReST was initially designed to be a closed, sane environment, but as additional packages have been developed it has become clear that this may not be desirable for some users. Future versions of ReST will make it easier to integrate ReST into their default shell environment. This will also make interpackage dependencies easier to handle.

7. *Persistent Groups* - Midway through development of the ReST Installer the purpose of Logical Groups changed from just a loose logical grouping of machines to a more tightly coupled group. Persistent groups would maintain the integrity of groups between installations through various methods.

8. *Support Windows as a target location* - Currently the ReST Installer requires a SSH daemon and a bash environment, neither by default available on Windows. In the future, a Windows communications plug-in could be researched to discover the viability of creating a plugin for compiling source based software on Windows.

9. *NetBuild Integration* - To help package makers simplify the library dependencies of their software, future versions of the ReST Installer may provide support for NetBuild.

10. *RIB Integration* - Future versions of ReST may be able to serve as a front-end for adding information to a Repository In a Box (RIB) registry and give an interface for viewing information already stored in RIB.

Conclusions

If filling a need is any indication of the future of a project, the Remote Software Toolkit has a very bright future. Many source based projects needing a simple means of gaining wider acceptance, until now, have had none. ReST empowers both developers and users to just use their software and in others to it's fullest potential. Besides all that, ReST is one of the coolest projects ever!

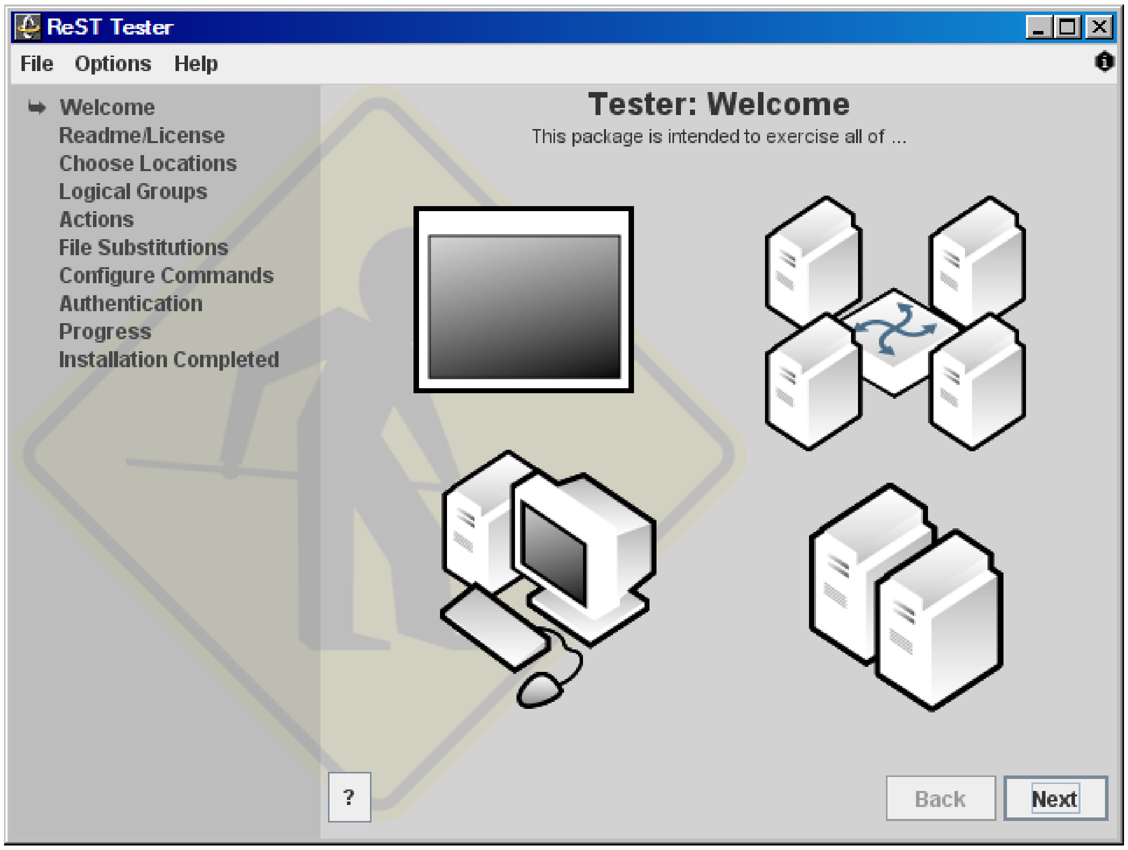
References

NetBuild - http://icl.cs.utk.edu/netbuild/

Repository In a Box (RIB) - http://icl.cs.utk.edu/rib/

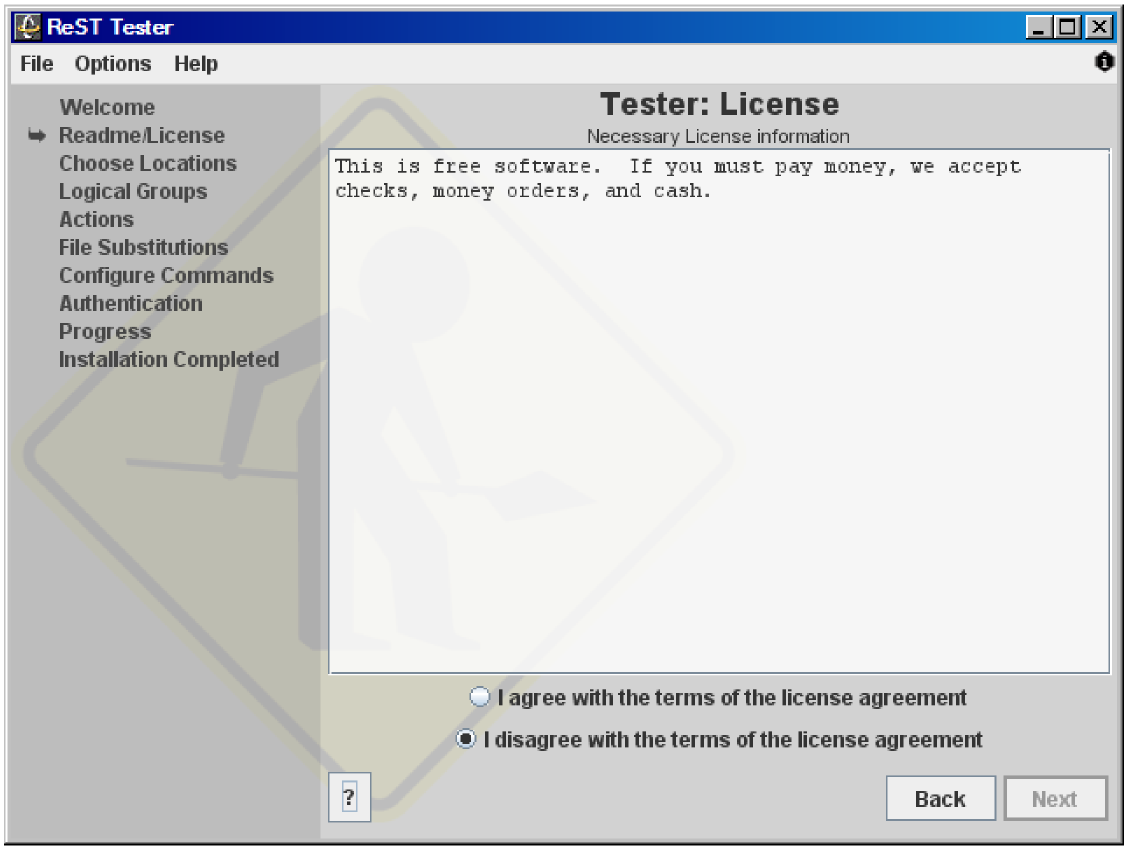
A. Screenshots - An Installer Walk Through

Figure A.1. Installer Welcome



Installer right after opening a package.

Figure A.2. License Support (Optional)

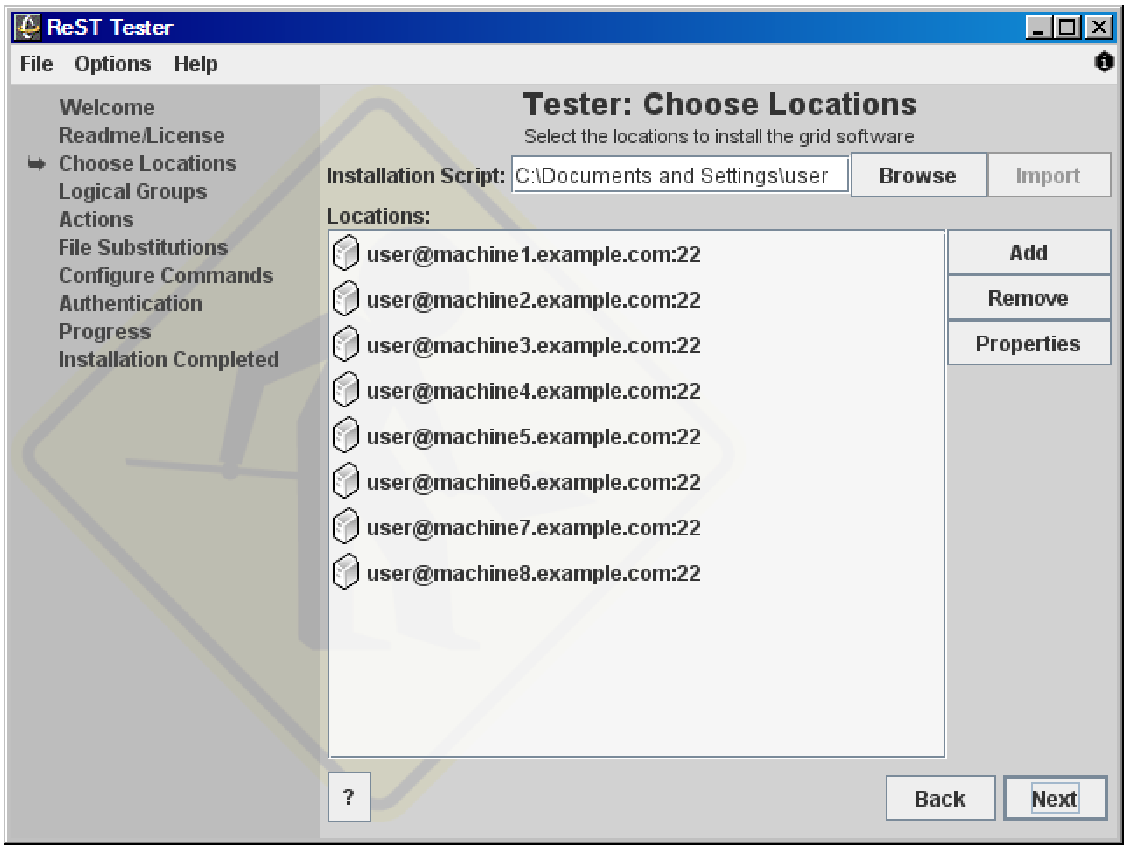


```
This is free software.  If you must pay money, we accept
checks, money orders, and cash.
```

I agree with the terms of the license agreement [✓ I disagree with the terms of the license agreement]

The License panel with the package specifing the license must be accepted. Notice the *Next* button is disabled until the license is accepted. The Readme display is exactly the same except without the acceptance/reject part of the panel.

Figure A.3. Setup Locations



Tester: Choose Locations

Select the locations to install the grid software

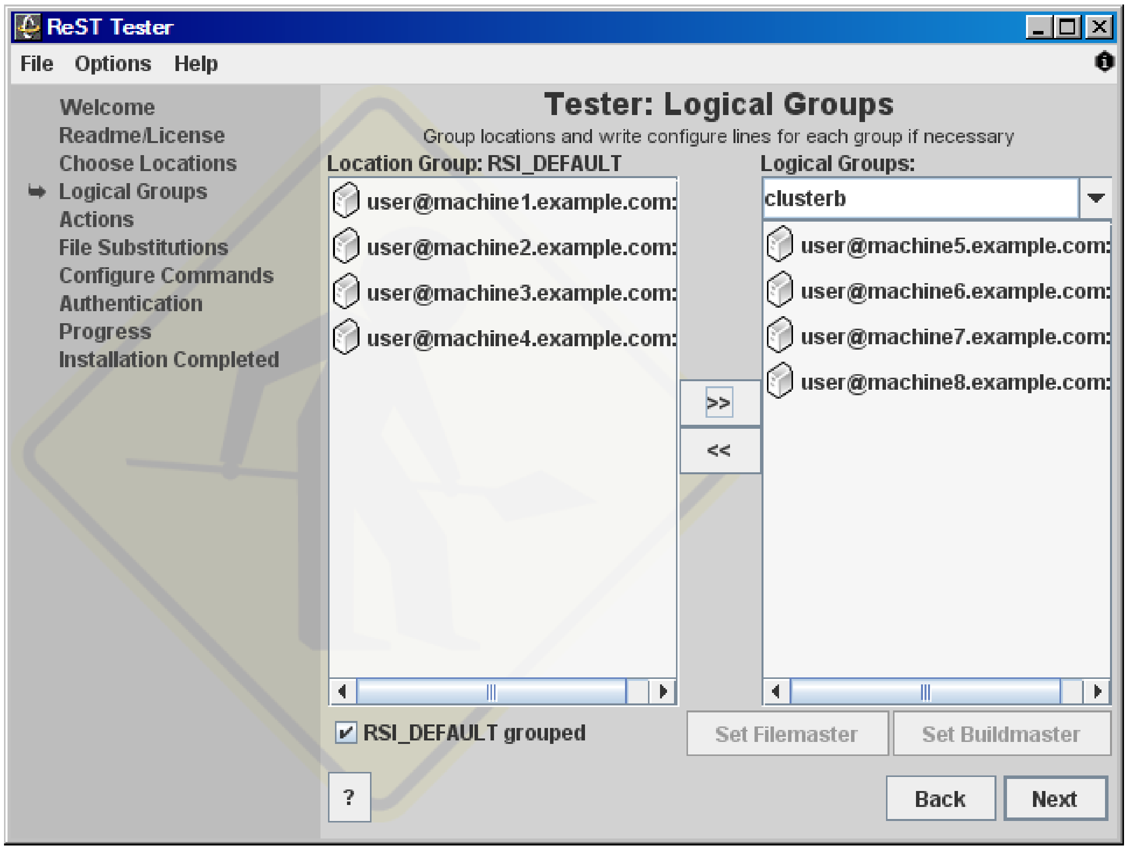
Installation Script: C:\Documents and Settings\user

Locations:

user@machine1.example.com:22
user@machine2.example.com:22
user@machine3.example.com:22
user@machine4.example.com:22
user@machine5.example.com:22
user@machine6.example.com:22
user@machine7.example.com:22
user@machine8.example.com:22

This screenshot shows the setup locations dialog of the ReST Installer. This user has setup 8 locations on which the software should be installed.

Figure A.4. Setup Logical Groups



Tester: Logical Groups

Group locations and write configure lines for each group if necessary

Location Group: RSI_DEFAULT

user@machine1.example.com:
user@machine2.example.com:
user@machine3.example.com:
user@machine4.example.com:

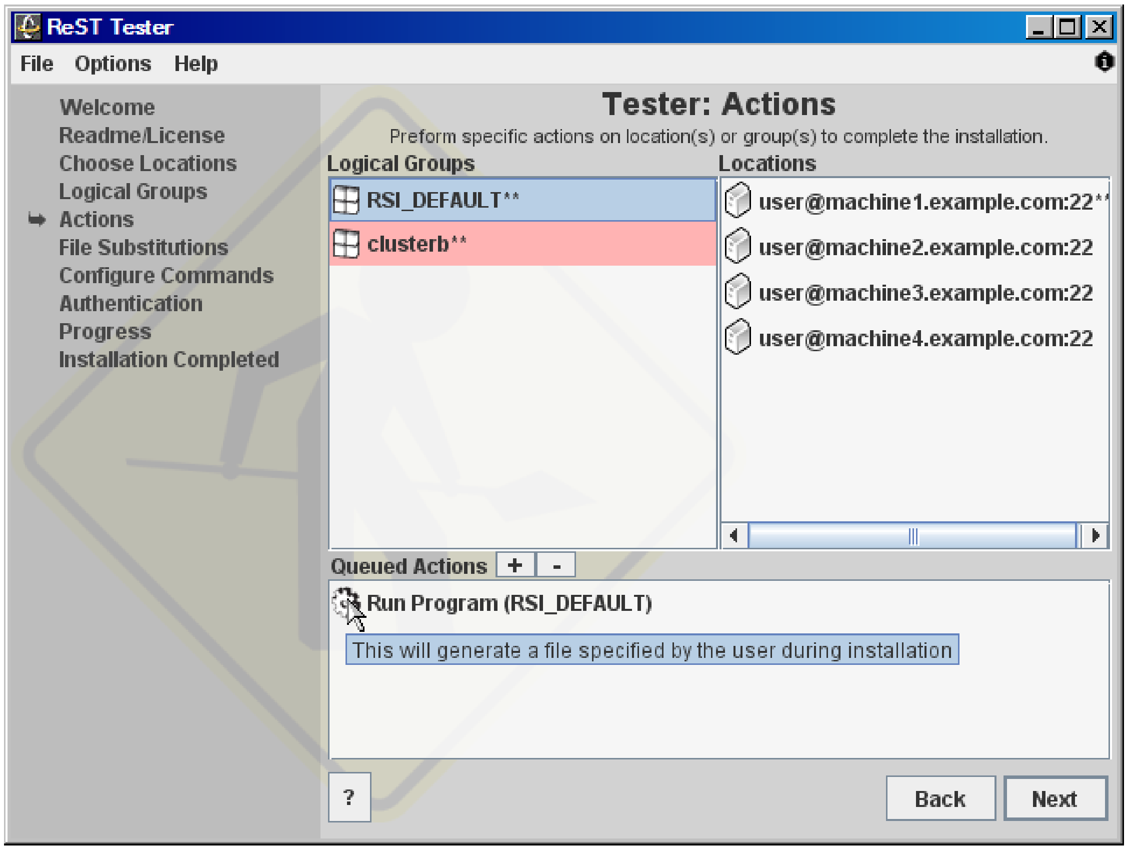
Logical Groups: clusterb

user@machine5.example.com:
user@machine6.example.com:
user@machine7.example.com:
user@machine8.example.com:

[✓] RSI_DEFAULT grouped

This screenshot shows the setup logical groups dialog of the ReST Installer. This user has split the 8 machines into 2 groups by selecting a group for 4 machines, while leaving the remaining machines in the RSI_DEFAULT.

Figure A.5. Setup Actions



Tester: Actions

Preform specific actions on location(s) or group(s) to complete the installation.

Logical Groups
- RSI_DEFAULT**
- clusterb**

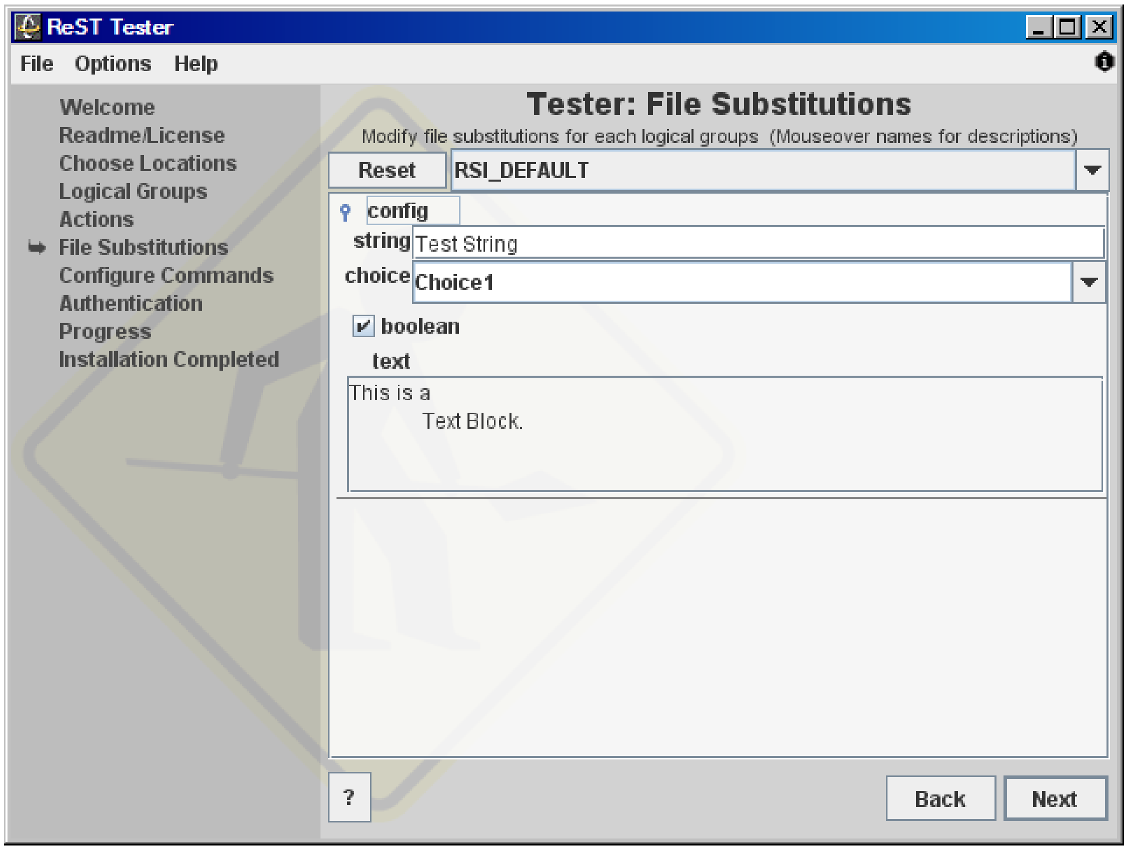
Locations
- user@machine1.example.com:22**
- user@machine2.example.com:22
- user@machine3.example.com:22
- user@machine4.example.com:22

Run Program (RSI_DEFAULT)

This will generate a file specified by the user during installation

This screenshot shows the setup actions dialog of the ReST Installer. This user has added the *Run Program* action to the RSI_DEFAULT group.

Figure A.6. Setup File Substitutions (Advanced)



Tester: File Substitutions

Modify file substitutions for each logical groups (Mouseover names for descriptions)

RSI_DEFAULT

config

string: Test String

choice: Choice1

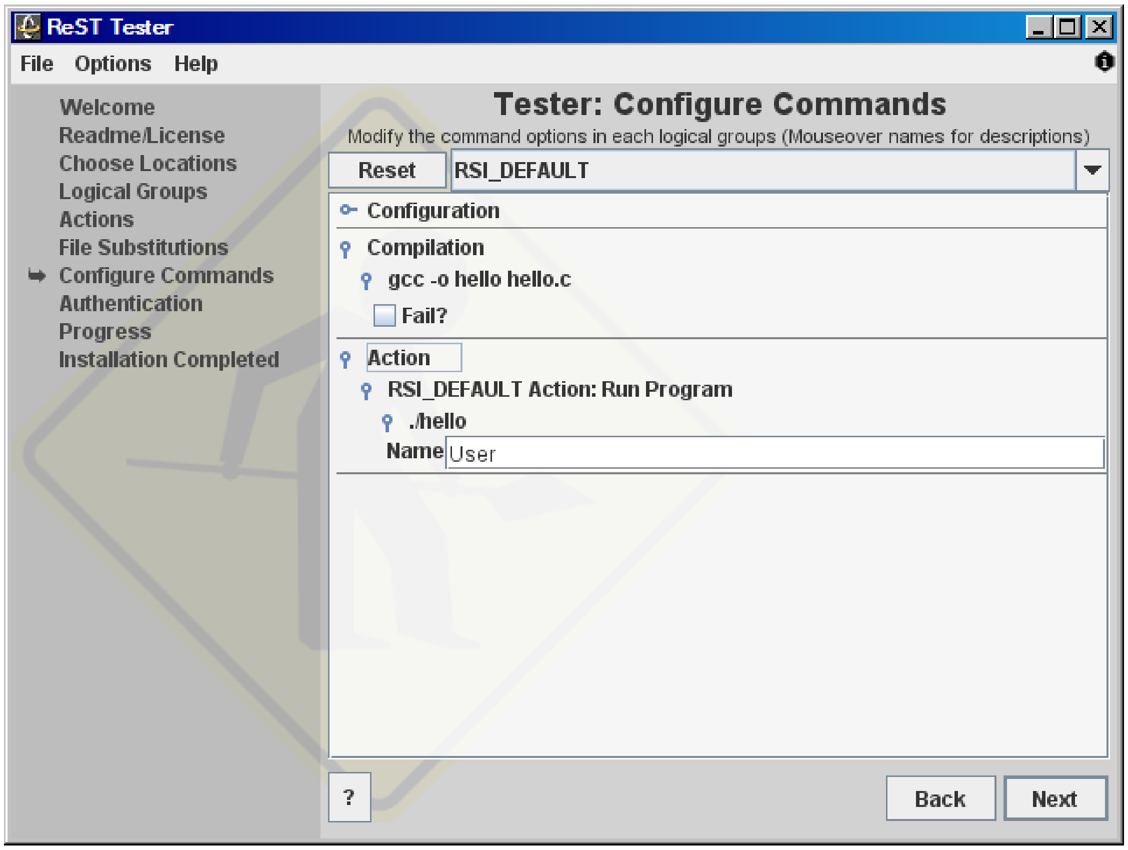
☑ boolean

text

```
This is a
        Text Block.
```

The File Substitutions panel is an advance panel and will only be shown if the package requires it or the *Advanced* item is selected from the *Options* menu. Each configurable file is configured on a per group (selected from the groups drop down menu) basis and is displayed in a hierarchial form within the panel.

Figure A.7. Setup Commands (Advanced)



Tester: Configure Commands

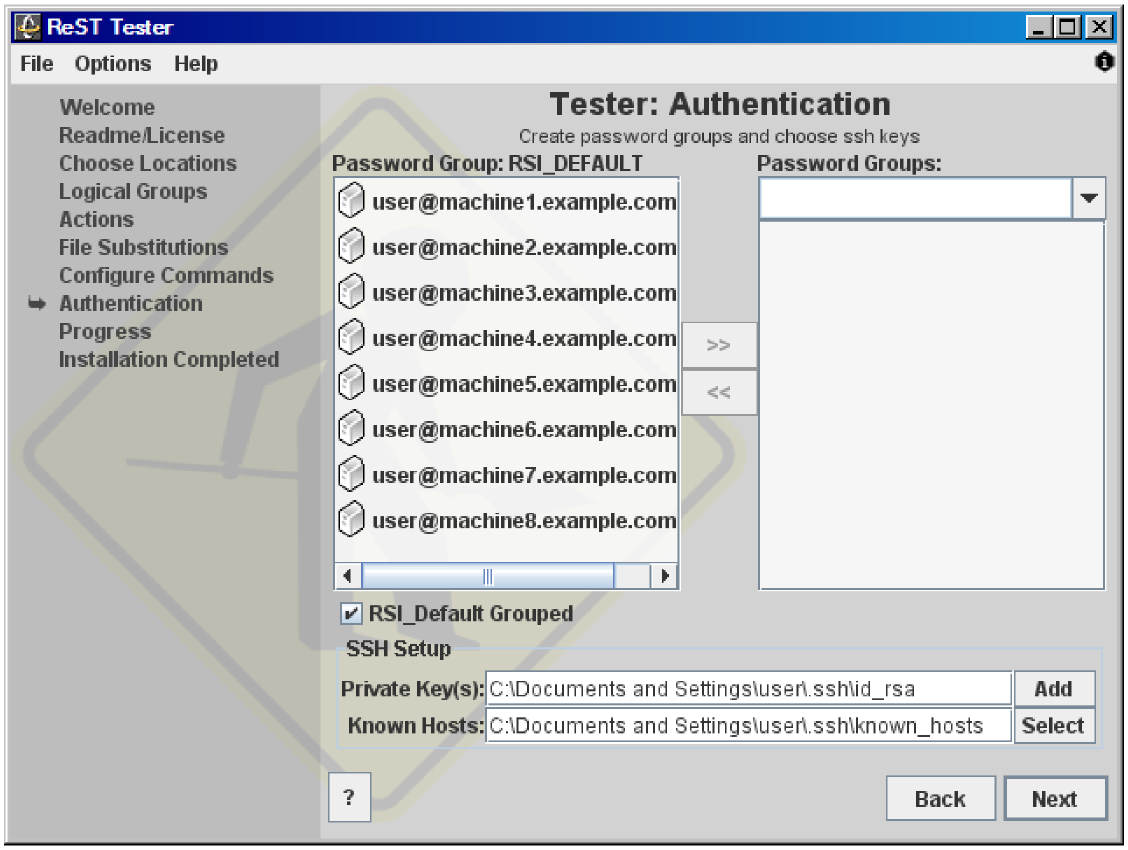
Modify the command options in each logical groups (Mouseover names for descriptions)

RSI_DEFAULT

Configuration
Compilation
gcc -o hello hello.c
Fail?
Action
RSI_DEFAULT Action: Run Program
./hello
Name User

The Configure Command panel is an advance panel and will only be shown if the package requires it or the *Advanced* item is selected from the *Options* menu. Each configurable command/action per group (selected from the groups drop down menu) is displayed in a hierarchial form within the panel.

Figure A.8. Setup Authentication



Tester: Authentication

Create password groups and choose ssh keys

Password Group: RSI_DEFAULT

user@machine1.example.com
user@machine2.example.com
user@machine3.example.com
user@machine4.example.com
user@machine5.example.com
user@machine6.example.com
user@machine7.example.com
user@machine8.example.com

Password Groups:

✓ RSI_Default Grouped

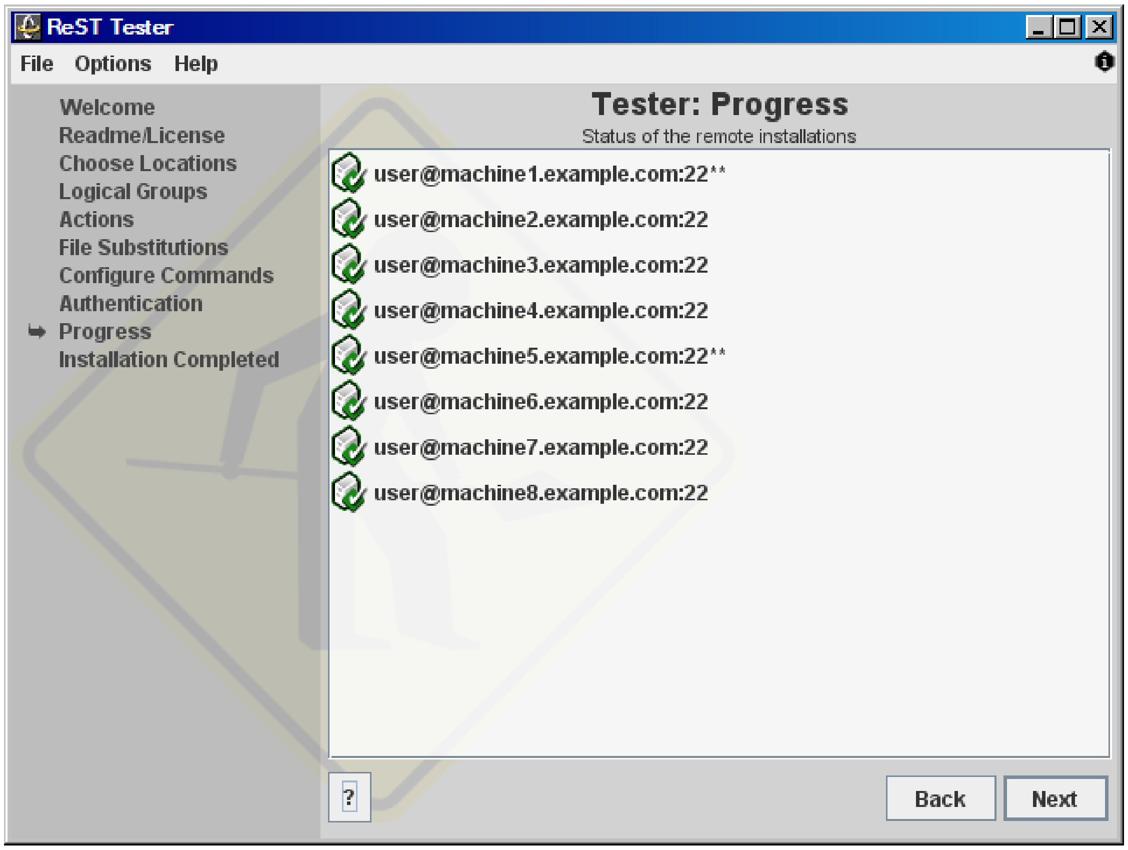
SSH Setup

Private Key(s): C:\Documents and Settings\user\.ssh\id_rsa

Known Hosts: C:\Documents and Settings\user\.ssh\known_hosts

The setup authentication panel of the ReST Installer. This user has provided the location of an SSH key and known_hosts file. Since the all the locations have been left in the default password group "grouped" if a password is required it will be used for all of the locations.

Figure A.9. Setup Installation



Tester: Progress

Status of the remote installations

user@machine1.example.com:22**
user@machine2.example.com:22
user@machine3.example.com:22
user@machine4.example.com:22
user@machine5.example.com:22**
user@machine6.example.com:22
user@machine7.example.com:22
user@machine8.example.com:22

The installation panel after completetion.