Red Hat Network Satellite 5.4

Getting Started Guide

Provisioning and Deployment with Red Hat Network Satellite

Edition 1

Last Updated: 2017-09-20

Red Hat Network Satellite 5.4 Getting Started Guide

Provisioning and Deployment with Red Hat Network Satellite
Edition 1

Landmann
rlandmann@redhat.com

Legal Notice

Abstract

This document contains information and instructions for use of the kickstart provisioning functionality in Red Hat Network Satellite. For more about Satellite basics, see the Satellite User Guide.

Table of Contents

CHAPTER 1. INTRODUCTION **3**

CHAPTER 2. KICKSTART **5**
2.1. REQUIRED PACKAGES 5
2.2. KICKSTART TREES 5
2.3. KICKSTART PROFILES 7
2.4. TEMPLATING 10
2.5. KICKSTARTING A MACHINE 13

CHAPTER 3. MULTIPLE SATELLITES **18**
3.1. INTER-SATELLITE SYNCHRONIZATION 18
3.2. ORGANIZATIONAL SYNCHRONIZATION 19
3.3. ISS USE CASES 20

CHAPTER 4. ADVANCED COMMANDS **23**
4.1. THE XML-RPC API 23
4.2. COBBLER 24
4.3. KOAN 28

CHAPTER 5. TROUBLESHOOTING **30**
5.1. Web Interface 30
5.2. Anaconda 30
5.3. Cobbler and Koan 31
5.4. Tracebacks 32
5.5. Registration 32
5.6. Kickstarts and Snippets 33

APPENDIX A. REVISION HISTORY **35**

CHAPTER 1. INTRODUCTION

To use the new provisioning functionality, you need one or more *target* machines. The target machines can be either physical, bare metal systems, or virtual machines. If you want to use RHN Satellite's virtual machine provisioning functionality, create the virtual machines using either Xen or KVM.

Definitions

Some terms used throughout this book:

Provisioning

The process of configuring a machine (physical or virtual) to a predefined known state. Satellite provisions all systems using the *kickstart* process.

Kickstarting

The process of installing a Red Hat system in an automated manner requiring little or no user intervention. Technically, *kickstart* refers to a mechanism in the Anaconda installation program that allows you to supply a concise description of the contents and configuration of a machine to the installer, which it then acts on. Such a concise system definition is referred to as a *Kickstart profile*.

Kickstart profile

The kickstart file is a text file that specifies all of the options needed to kickstart a machine, including partitioning information, network configuration, and packages to install. In RHN Satellite, a Kickstart Profile is a superset of a traditional Anaconda kickstart definition, as Satellite's implementation builds on Cobbler's enhancements to kickstarting. A Kickstart Profile presumes the existence of a Kickstart Tree.

Kickstart Tree

The software and support files needed in order to kickstart a machine. This is also often called an "install tree". This is usually the directory structure and files pulled from the installation media that ships with a particular release. In Cobbler terminology, a Kickstart Tree is part of a distribution.

PXE (Preboot eXecution Environment)

A low-level protocol that makes it possible to kickstart bare-metal machines (usually physical, or *real*, machines) on power-up with no pre-configuration of the target machine itself. PXE relies on a DHCP server to inform clients about bootstrap servers (for purposes of this document, Satellite 5.3.0 or greater installations). PXE must be supported in the firmware of the target machine in order to be used. It is possible to use the virtualization and reinstall facilities of Satellite without PXE, though PXE is very useful for booting new physical machines, or reinstalling machines that are not registered to Satellite.

Provisioning Scenarios

The types of provisioning scenarios supported by RHN Satellite:

New installations

Provisioning a system that has not previously had an operating system installed (also known as *bare metal* installations).

Virtual installations

Satellite supports KVM, Xen fully-virtualized guests, and Xen para-virtualized guests.

Re-provisioning

Both physical and guest systems can be re-provisioned, provided that they have been registered to the same Satellite instance. See Section 2.5.2, “Reprovisioning”.

CHAPTER 2. KICKSTART

2.1. REQUIRED PACKAGES

If you are using a custom distribution, you will require the following packages, which are available from any **rhn-tools** Red Hat Network channel:

- **koan**
- **spacewalk-koan**

It is advisable to clone an existing **rhn-tools** channel in order to have access to these packages from your custom channel.

RHN Satellite expects the **kernel** and **initrd** files to be in specific locations within the kickstart tree. However, these locations are different for different architectures. The table below explains the different locations:

Table 2.1. Required Distribution Files by Architecture

Architecture	kernel	Initial RAM Disk image
IBM s390x	*TREE_PATH*/images/kernel.img	*TREE_PATH*/images/initrd.img
PowerPC	*TREE_PATH*/ppc/ppc64/vmlinuz	*TREE_PATH*/images/pxeboot/vmlinux
All other architectures	*TREE_PATH*/images/pxeboot/vmlinuz	*TREE_PATH*/images/pxeboot/initrd.img

2.2. KICKSTART TREES

You must have at least one kickstart tree installed on your Satellite in order to use kickstart provisioning. Kickstart trees can be installed either automatically or manually.

Procedure 2.1. Installing Kickstart Trees Automatically

For all distributions that have a base channel in Red Hat Network, kickstart trees can install automatically. This occurs as part of normal channel synchronization via **satellite-sync**.

1. Choose which distribution you would like to base your kickstarts on and locate that distribution's base channel, and its corresponding RHN Tools channel.

   For example, if you want to use Red Hat Enterprise Linux 5 with x86 architecture, you will require the **rhel-i386-server-5** channel and its corresponding RHN Tools channel **rhn-tools-rhel-i386-server-5**.

2. If you are using a connected Satellite, synchronize your Satellite with the Red Hat servers directly with the **satellite-sync** command. If your Satellite is disconnected, you will need to obtain disconnected channel dumps from the Red Hat servers and synchronize with those.

3. Synchronizing the channel will automatically create a corresponding kickstart tree for that distribution.

Procedure 2.2. Installing Kickstart Trees Manually

To kickstart a custom distribution, a distribution not supported by Red Hat, or a beta version of Red Hat Enterprise Linux, you will need to create the corresponding kickstart tree manually. You will require an installation ISO for the distribution you are kickstarting.

1. Copy the installation ISO to your satellite server and mount it to **/mnt/iso**

2. Copy the contents of the ISO to a custom location. It is recommended that you create a directory within **/var/satellite** for all of your custom distributions. For example, you might copy a RHEL beta distribution's contents to **/var/satellite/custom-distro/rhel-i386-server-5.3-beta/**

3. Use the RHN Satellite web interface to create a custom software channel. Use **Channels** → **Manage Software Channels** → **Create New Channel** to create a parent channel with an appropriate name and label. For the example used above, you might use the label **rhel-5.3-beta**.

4. Push the software packages from the tree location to the newly created software channel using the **rhnpush** command:

```
rhnpush --server=http://localhost/APP -c 'rhel-5.3-beta' \  -d
/var/satellite/custom-distro/rhel-i386-server-5.3-beta/Server/
```

   The sub-directory within the tree could be different depending on your distribution.

5. Once the software packages have been pushed, they can be deleted from within the tree path using the **rm** command. The packages are still stored on the Satellite server within the channel, and are no longer required in the tree.

```
rm /var/satellite/custom-distro/rhel-i386-server-5.3-
beta/Server/*.rpm
```

NOTE

You can choose to leave the software packages within the kickstart tree. This will allow them to be installed with the **yum** command at any time later on.

6. Use the RHN Satellite web interface to create the distribution. Use **Systems** → **Kickstart** → **Distributions** → **create new distribution** to create the distribution, using an appropriate label and the full tree path (such as **/var/satellite/custom-distro/rhel-i386-server-5.3-beta/**. Select the base channel you created earlier, and the correct Installer Generation (such as **Red Hat Enterprise Linux 5**). To complete the creation, select **Create Kickstart Distribution**.

7. To maintain the same software across multiple environments and systems, the RHN Tools child channel from an existing Red Hat Enterprise Linux base channel can be cloned as a child channel of the newly created base channel. Cloning a child channel can be done by:

   1. On the Satellite web interface, click on **Channels** → **Manage Software Channels** → **Clone Channel**

   2. Choose the Child Channel you wish to clone from the drop down box **Clone From:** and choose the clone state.

3. Click **Create Channel**.

4. Fill in the necessary information and choose the parent channel that the cloned child channel needs to be under.

5. Click **Create Channel**.

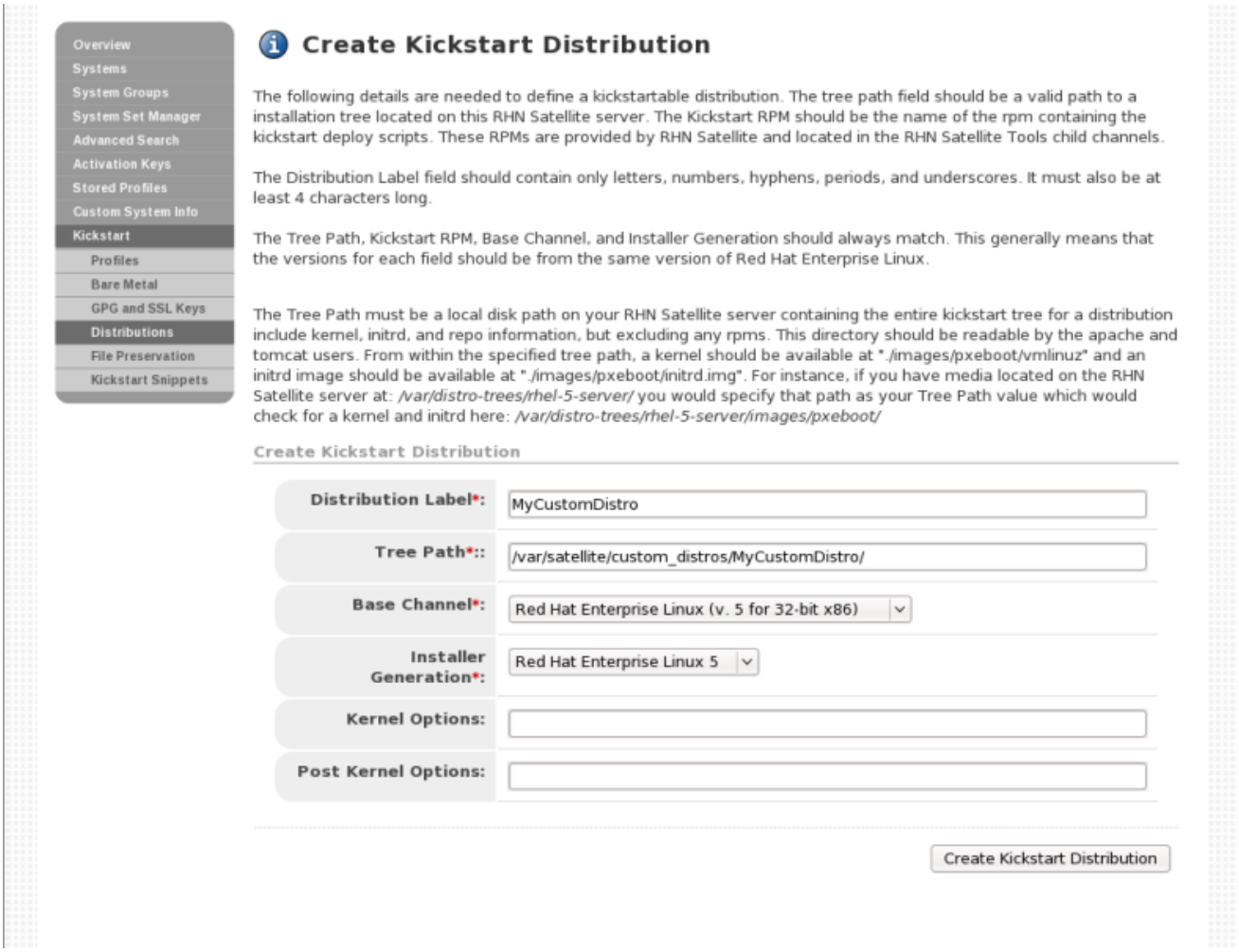


Figure 2.1. Creating a Kickstart Distribution

2.3. KICKSTART PROFILES

Kickstart profiles specify the configuration options to be used for the installation.

Kickstart profiles can be created using a *wizard* interface, which generates a profile based on the answers you give to a series of questions. They can also be created using the *raw method*, which gives you complete control over the contents of the profile.

Procedure 2.3. Creating a Kickstart Profile with a Wizard

1. Select **Systems** → **Kickstart** → **Create a New Kickstart Profile**

2. Provide an appropriate **Label**, and select the desired **Base Channel** and **Kickstartable Tree**

3. Select the desired **Virtualization Type**. See Virtualization Types for more information about virtualization types. Click **next** to continue.

4. Select the download location for the kickstart profile. If you are using a custom distribution, enter the location of its tree as a URI (both HTTP and FTP are supported), otherwise, use the default option. Click **next** to continue.

5. Enter the root password and click **finish** to complete the profile creation.

6. The complete kickstart profile will be created. View the profile by clicking **Kickstart File**.

Procedure 2.4. Creating a Kickstart Profile with the Raw Method

1. Select **Systems** → **Kickstart** → **Upload a New Kickstart File**

2. Provide an appropriate **label**, and select the desired **distribution**

3. Select the desired **Virtualization Type**. See Virtualization Types for more information about virtualization types.

4. If you have an existing kickstart profile, upload the file. Otherwise, write the kickstart profile in the **File Contents** text box.

   Here is a sample raw kickstart that can be used as a starting point:

```
install
text
network --bootproto dhcp
url --url http://$http_server/ks/dist/org/1/ks-rhel-i386-server-5
lang en_US
keyboard us
zerombr
clearpart --all
part / --fstype=ext3 --size=200 --grow
part /boot --fstype=ext3 --size=200
part swap --size=1000   --maxsize=2000
bootloader --location mbr
timezone America/New_York
auth --enablemd5 --enableshadow
rootpw --iscrypted $1$X/CrCfCE$x0veQO88TCm2VprcMkH.d0
selinux --permissive
reboot
firewall --disabled
skipx
key --skip

%packages
@ Base

%post
$SNIPPET('redhat_register')
```

5. The RHN Satellite server does not handle the specified distribution as the ***url*** in the kickstart, so you will need to include the ***url --url*** option in your profile, similar to the following:

```
url --url http://satellite.example.com/ks/dist/org/1/my_distro
```

   Replace **my_distro** with the distribution label and **1** with your org id.

6. Raw kickstart profiles use ***$http_server*** instead of the Satellite's host name. This will be filled in automatically when the kickstart template is rendered.

7. The ***redhat_register*** snippet is used to handle registration.

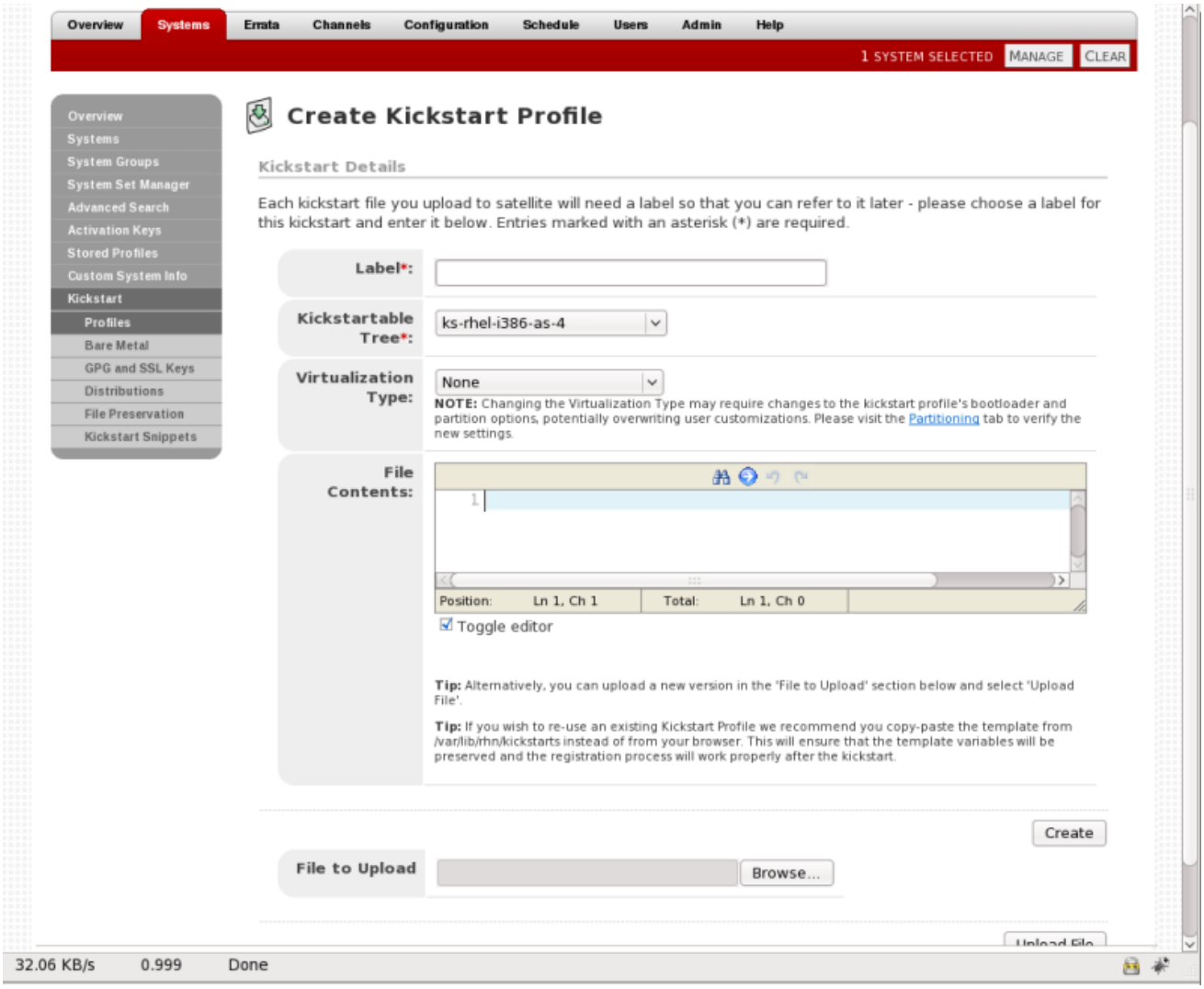


Figure 2.2. Raw Kickstart

Virtualization Types

All kickstart profiles have a virtualization type associated with them. This table outlines the different options:

Table 2.2. Virtualization Types

Type	Description	Uses
none	No virtualization	Use this type for normal provisioning, bare metal installations, and virtualized installation that are not Xen or KVM (such as VMware, or Virtage)
KVM Virtualized Guest	KVM guests	Use this type for provisioning KVM guests

Type	Description	Uses
Xen Fully-Virtualized Guest	Xen guests	Use this type for provisioning Xen guests  **NOTE** This option requires hardware support on the host, but does not require a modified operating system on the guest.
Xen Para-Virtualized Guest	Xen Guests	Use this type for provisioning a virtual guest with Xen para-virtualization. Para-virtualization is the fastest virtualization mode. It requires a PAE flag on the system CPU, and a modified operating system. Red Hat Enterprise Linux 5 supports guests under para-virtualization.
Xen Virtualization Host	Xen hosts	Use this type for provisioning a virtual host with Xen para-virtualization. Xen para-virtualized guests and hosts are supported, if the hardware is compatible.

Kickstart profiles created to be used as Xen hosts must include the **kernel-xen** package in the ***%packages*** section.

Kickstart profiles created to be used as KVM hosts must include the **qemu** package in the ***%packages*** section.

Fully virtualized systems may require virtualization support to be turned on in the computer's BIOS menu.

NOTE

For more information about kickstart, see the *Kickstart Installations* chapter in the *Red Hat Enterprise Linux Installation Guide*.

2.4. TEMPLATING

Kickstart *templating* allows you to include variables, snippets, and flow control statements such as **for** loops and **if** statements in your kickstart files. This is achieved using the **cheetah** tool.

There are a variety of reasons why templating might be useful:

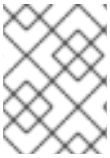
- Reusing a particular section of a kickstart, such as a disk partitioning section, between multiple kickstarts.
- Performing some **%post** actions consistently across multiple kickstarts.
- Defining a snippet across multiple server roles such as DNS server, proxy server, and web server. For example, a web server might have the following snippet defined:

```
httpd
mod_ssl
mod_python
```

  To create a web server profile, include the web server snippet in the **%package** section of your kickstart file. For a profile to be both a web server and a proxy server, include both snippets in the package section. To add another package to the web server snippet, **mod_perl** for example, update the snippet, and all profiles that are using that snippet are dynamically updated.

Variables

Templating allows you to define a variable to be used throughout a kickstart file. Variables are subject to a form of inheritance that allows them to be set at one level and overridden at levels below them. So, if a variable is defined at the system level, it will override the same variable defined at the profile or kickstart tree levels. Likewise, if a variable is defined at the profile level, it will override the same variable defined at the kickstart tree level.



NOTE

Note that kickstart tree variables cannot be defined for automatically generated kickstart trees, such as those created when you perform a satellite synchronization.

Snippets

Snippets reuse pieces of code between multiple kickstart templates. They can span many lines, and include variables. They can be included in a kickstart profile by using the text **$SNIPPET('snippet_name')**. You can make a snippet for a package list, for a **%post** script, or for any text that would normally be included in a kickstart file.

To manage snippets navigate to **Systems** → **Kickstart** → **Kickstart Snippets**.

The **Kickstart Snippets** page displays several default snippets that cannot be edited, but are available to be used by any organization. Default snippets can be used in kickstarts that have been either written on or uploaded to the RHN Satellite server. Default snippets are stored on the RHN Satellite server's file system in **/var/lib/cobbler/snippets/**. There is a template from a wizard-style kickstart located in **/var/lib/rhn/kickstarts/wizard/**, which explains the different default snippets and how they are used.

The **redhat_register** snippet is a default snippet that is used to register machines to a RHN Satellite server as part of a kickstart. It uses a variable called **redhat_management_key** to register the machine. To use the snippet, set the **redhat_management_key** variable at either the system, profile, or distribution level and then add **$SNIPPET('redhat_register')** to a **%post** section of the kickstart. Any wizard-style kickstarts that are generated by the RHN Satellite server will already include this snippet in the **%post** section.

The **Custom Snippets** tab allows you to view and edit snippets created for use by your organization. New snippets can be created by clicking **create new snippet**. Custom snippets are stored in the **/var/lib/rhn/kickstarts/snippets/** directory. RHN Satellite stores snippets for different organizations in different directories, so custom snippets will be stored with a filename similar to the following, where *1* is the organization ID:

```
$SNIPPET('spacewalk/1/snippet_name')
```

To determine the text to use to insert the snippet in the kickstart, look for the **Snippet Macro** column on the snippet list, or on the snippet details page.

NOTE

Snippets exist at a global level and do not share the same inheritance structure as variables. However, variables can be used within snippets to change the way they behave when different systems request a kickstart.

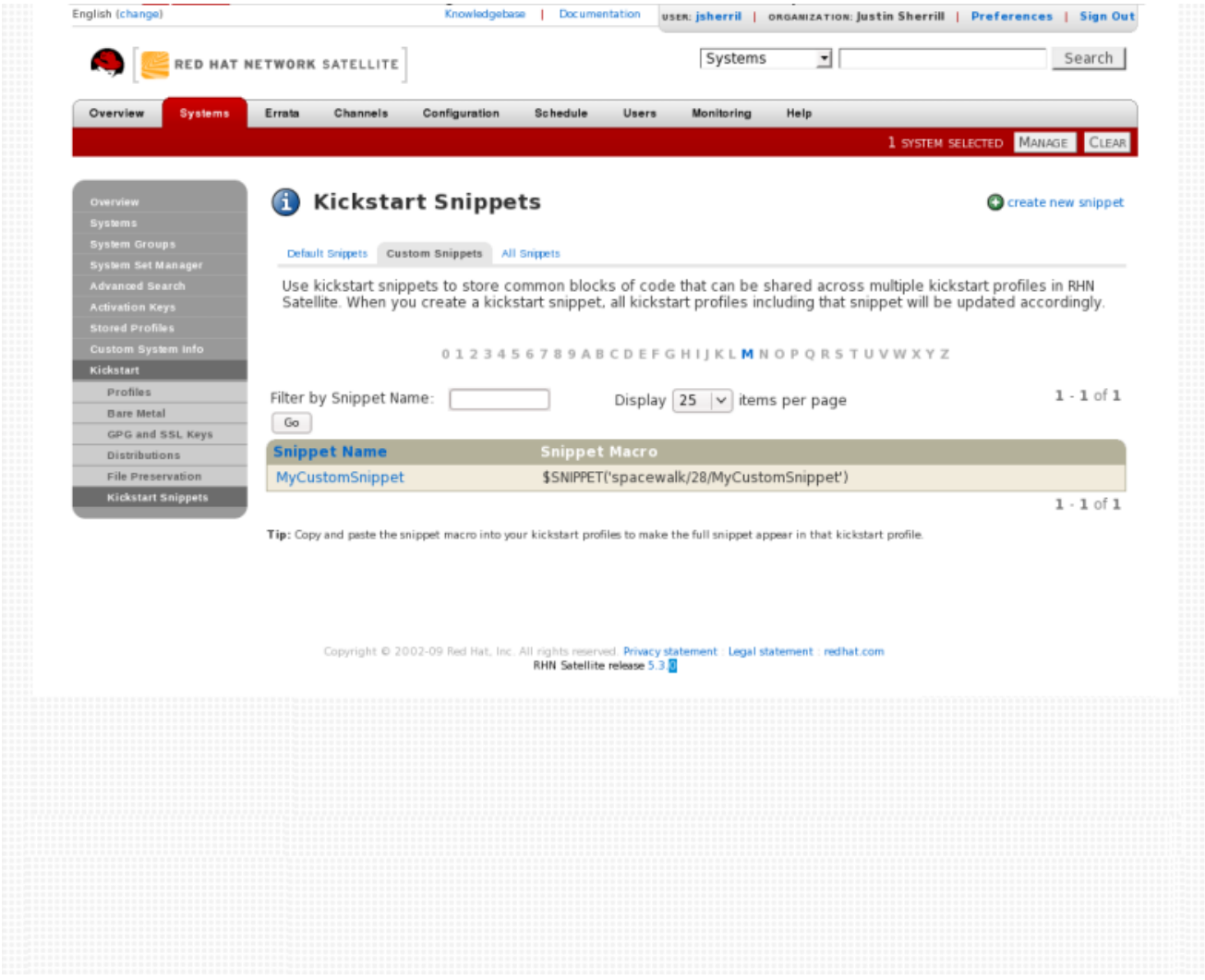


Figure 2.3. Kickstart Snippets

Escaping Special Characters

The *$* and *#* characters are used during templating for specifying variables and control flow. If you need to these characters for any other purpose in a script, they will need to be escaped so that they are not recognized as a variable. This can be achieved in several ways:

- Placing a backslash character (`\`) before every instance of `$` or `#` that you want to be ignored during templating.
- Wrap the entire script in `#raw ... #end raw`

  All `%pre` and `%post` scripts created using the wizard-style kickstarts are wrapped with `#raw...#end raw` by default. This can be toggled using the `Template` checkbox available when editing a `%post` or `%pre` script.
- Include `#errorCatcher Echo` in the first line of the snippet.

Example 2.1. Escaping Special Characters in templates

This example describes how to escape special characters in kickstart templates.

The following bash script needs to be inserted in a `%post` section:

```
%post
echo $foo > /tmp/foo.txt
```

Without the `$` being escaped, the templating engine will try to find a variable named `$foo` and would fail because `foo` does not exist as a variable.

The simplest way to escape the `$` is by using a backslash character (`\`):

```
%post
echo \$foo > /tmp/foo.txt
```

This will cause `\$foo` to be rendered as `$foo`.

A second method is to wrap the entire bash script in `#raw ... #end raw`, as follows

```
%post
#raw
echo $foo > /tmp/foo.txt
#end raw
```

The final method is to include `#errorCatcher Echo` in the first line of the kickstart template. This instructs the templating engine to ignore any variables that do not exist and print out the text as is. This option is already included in the wizard-style kickstarts, and can be includes in any raw kickstarts you create manually.

2.5. KICKSTARTING A MACHINE

2.5.1. Kickstarting from Bare Metal

When a machine has no existing operating system or has the wrong operating system installed, it is referred to as a *bare metal* machine. There are three ways to provision a machine from bare metal:

- Standard operating system installation media
- PXE boot

- Cobbler boot disk. For more information on Cobbler, see Section 4.2, “Cobbler”

Procedure 2.5. Booting from Installation Media

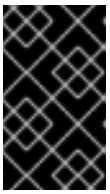
1. Insert installation media into the machine. The media must match the kickstart you intend to use. For example, if the kickstart is configured to use the **ks-rhel-i386-server-5-u2** kickstart tree, use Red Hat Enterprise Linux 5.2 i386 installation media.
2. When you are given a boot prompt, activate the kickstart by giving this command:

```
linux ks=http://satellite.example.com/path/to/kickstart
```

3. The system will boot, download the kickstart, and install automatically.

Procedure 2.6. PXE Booting

In order to be able to perform a PXE boot, each system you have must support PXE booting at the BIOS level. Nearly all recent hardware should be able to do this. Additionally, you must have a DHCP server, even if your systems are to be configured statically after installation.

1. 

   IMPORTANT

   If you have a DHCP server deployed on another system on the network, you will need administrative access to the DHCP server in order to edit the DHCP configuration file.

   If your machines reside on multiple networks, you will need to make certain that all of your machines can connect to the DHCP server. This can be achieved by multi-homing your DHCP server (using either a real or trunked VLAN) and configuring any routers or switches to pass the DHCP protocol across network boundaries.

   Configure your DHCP server so that it points to the PXE server by setting the ***next-server*** address for the systems you want to be managed by RHN Satellite.

   Configure your DHCP server to point to your domain and IP addresses, by including the following lines. This will allow it to use hostnames to perform the installation:

```
option domain-name DOMAIN_NAME;
option domain-name-servers IP_ADDRESS1, IP_ADDRESS2;
```

2. On the DHCP server, switch to the root user and open the **/etc/dhcpd.conf** file. Append a new class with options for performing PXE boot installation:

```
allow booting;
allow bootp;
class "PXE" {
  match if substring(option vendor-class-identifier, 0, 9) =
"PXEClient";
  next-server 192.168.2.1;
  filename "pxelinux.0";
}
```

   This class will perform the following actions:

1. Enable network booting with the **bootp** protocol
2. Create a class called ***PXE***. If a system is configured to have PXE first in its boot priority, it will identify itself as **PXEClient**.
3. The DHCP server directs the system to the Cobbler server at the IP address *192.168.2.1*
4. The DHCP server refers to the boot image file at **/var/lib/tftpboot/pxelinux.0**

3. Configure Xinetd. Xinetd is a daemon that manages a suite of services including TFTP, the FTP server used for transferring the boot image to a PXE client.

   Enable Xinetd using the **chkconfig** command:

   ```
   chkconfig xinetd on
   ```

   Alternatively, switch to the root user and open the **/etc/xinetd.d/tftp** file. Locate the ***disable = yes*** line and change it to read **disable = no**.

4. Start the Xinetd service so that TFTP can start serving the **pxelinux.0** boot image:

   ```
   chkconfig --level 345 xinetd on
   /sbin/service xinetd start
   ```

   The **chkconfig** command turns on the **xinetd** service for all user runlevels, while the **/sbin/service** command turns on **xinetd** immediately.

2.5.2. Reprovisioning

Reinstalling an existing system is referred to as *reprovisioning*. Reprovisioning can be done using the RHN Satellite web interface, and the system will use the same system profile that it had before it was reprovisioned. This will preserve a lot of the information and settings about the system.

Reprovisioning can be scheduled from the **Provisioning** tab while viewing a system. To configure additional options, go to the **Advanced Configurations** page, which allows you to configure details such as kernel options, networking information, and package profile synchronization. The **Kernel Options** section provides access to the kernel options used during kickstart and **Post Kernel Options** are the kernel options that will be used after the kickstart is complete and the system is booting for the first time.

> **Example 2.2. Configuring Kernel Options and Post Kernel Options**
>
> This example describes the difference between kernel options and post kernel options in the reprovisioning configuration process.
>
> To establish a VNC connection to monitor the kickstart remotely, include **vnc vncpassword=PASSWORD** in the **Kernel Options** line.
>
> If you want the kernel of the resulting system to boot with the **noapic** kernel option, add **noapic** to the **Post Kernel Options** line.

Procedure 2.7. File Preservation

The *File Preservation* feature can be used to keep files from being lost during a reprovisioning. This feature stores files temporarily during the kickstart and restores them after the reprovisioning is complete.

NOTE

File preservation lists are only available on wizard-style kickstarts, and can only be used during reprovisioning.

1. Go to **Systems** → **Kickstart** → **File Preservation** → **create new file preservation list** and create a list of files to preserve.
2. Go to **Systems** → **Kickstart** → **Profiles** and associate the file preservation list with a kickstart by selecting the desired profile.
3. Go to **System Details** → **File Preservation** and select the file preservation list.

2.5.3. Virtualized Guest Provisioning

Virtual Guest Provisioning is supported in RHN Satellite 5.4.1 using the following virtualization technologies:

- KVM Virtualized Guest
- Xen Fully-Virtualized Guest
- Xen Para-Virtualized Guest

Procedure 2.8. Provisioning a Virtualized Guest

1. Check that the host system has a **`Virtualization`** or **`Virtualization Platform`** system entitlement
2. On the **`Systems`** page, select the appropriate virtual host, then select **Virtualization** → **Provisioning**. Select the appropriate kickstart profile and enter a guest name.
3. To configure additional parameters such as guest memory and CPU usage, click the **Advanced `Configuration`** button. This will allow you to configure:
   - Network: static or DHCP
   - Kernel options
   - Package profile synchronization: when the kickstart finishes the system will synchronize its package profile to that of another system or a stored profile
   - Memory allocation: RAM (Defaults to 512MB)
   - Virtual disk size
   - Virtual CPUs (Defaults to 1)
   - Virtual bridge: The networking bridge used for the install. Defaults to **`xenbr0`** for Xen provisioning, and **`virbr0`** for KVM.

NOTE

The **virbr0** networking bridge will not allow outside networking. If you require outside networking, configure the host to create an actual bridge instead. However, **xenbr0** is an actual bridge, and it is recommended that you use it if possible.

- Virtual storage path: Path to either a file, LVM Logical Volume, directory, or block device with which to store the guest's disk information, such as **/dev/sdb**, **/dev/LogVol00/mydisk**, **VolGroup00**, or **/var/lib/xen/images/myDisk**.

4. Click **Schedule Kickstart and Finish**

2.5.4. Provisioning Through an RHN Proxy

Provisioning can also be achieved using an Red Hat Network Proxy that has been installed and registered to RHN Satellite.

1. When provisioning a virtual guest or doing a reprovisioning of a system, select the desired proxy from the **Select Satellite Proxy** drop down box.

2. For a bare metal installation, replace the RHN Satellite's fully qualified domain name (FQDN) with that of the proxy's FQDN. For example if the URL to the kickstart file is:

   ```
   http://satellite.example.com/ks/cfg/org/1/label/myprofile
   ```

   Then to kickstart through the proxy, use:

   ```
   http://proxy.example.com/ks/cfg/org/1/label/myprofile
   ```

CHAPTER 3. MULTIPLE SATELLITES

Inter-satellite synchronization (ISS) allows you to coordinate content between Satellites. This feature can be used in several different ways, depending on the needs of your organization. This chapter contains a section on use cases, and how best to set up ISS to suit your organization.

ISS Requirements

In order to be able to use ISS, you will require:

- Two or more RHN Satellite servers
- At least one RHN Satellite populated with at least one channel
- For secure connections, each slave RHN Satellite will also require a master RHN Satellite SSL certificate

3.1. INTER-SATELLITE SYNCHRONIZATION

Procedure 3.1. Configuring the Master Server

The master server is used to determine what files will be synchronized to the other satellites.

1. Enable the inter-satellite synchronization (ISS) feature. Open the **/etc/rhn/rhn.conf** file, and add or amend the following line to read:

```
disable_iss=0
```

2. In the **/etc/rhn/rhn.conf** file, locate the **allowed_iss_slaves=** line. By default, no slave satellites are specified for synchronization. Enter the hostname of each slave satellite server, separated by commas:

```
allowed_iss_slaves=slave1.satellite.example.org,slave2.satellite.exa
mple.org
```

3. Save the configuration file, and restart the **httpd** service:

```
service httpd restart
```

Procedure 3.2. Configuring Slave Servers

Slave satellite servers are the machines that will have their content synchronized to the master server.

1. In order to securely transfer content to the slave servers, you will require the **ORG-SSL** certificate from the master server. You can download the certificate over HTTP from the **/pub/** directory of any satellite. The file is called **RHN-ORG-TRUSTED-SSL-CERT**, but can be renamed and placed anywhere in the local filesystem of the slave, such as the **/usr/share/rhn/** directory.

2. View the list of channels available for synchronization from the master server with the following command. This will display official Red Hat channels as well as any available custom channels:

```
satellite-sync --iss-parent=master.satellite.example.com --ca-
cert=/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT --list-channels
```

Replace *master.satellite.example.com* with the hostname of the master server.

Procedure 3.3. Performing an Inter-Satellite Synchronization

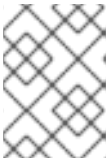
Once the master and slave servers are configured, you can perform a synchronization between them.

1. On the slave servers, open the **/etc/rhn/rhn.conf** file in your preferred text editor, and add the master server hostname and SSL certificate file path details:

```
iss_parent      = master.satellite.example.com
iss_ca_chain    = /usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT
```

2. Begin the synchronization by running the **satellite-sync** command:

```
satellite-sync -c your-channel
```



NOTE

Command line options you provide with the **satellite-sync** command will override any custom settings in the **/etc/rhn/rhn.conf** file.

3.2. ORGANIZATIONAL SYNCHRONIZATION

ISS can also be used to import content to any specific organization. This can be done locally or by using remote synchronization. This function is useful for a disconnected satellite with multiple organizations, where content is retrieved through channel dumps or by exporting from connected satellites and then importing it to the disconnected satellite. Organizational synchronization can be used to export custom channels from connected satellites. It can also be used effectively to move content between multiple organizations.

Organizational synchronization follows a clear set of rules in order to maintain the integrity of the source organization:

- If the source content belongs to the ***NULL*** organization (that is, it is Red Hat content) it will default to the ***NULL*** organization even if a destination organization is specified. This ensures that specified content is always in the privileged ***NULL*** organization.
- If an organization is specified at the command line, content will be imported from that organization.
- If no organization is specified, it will default to organization 1.

The following are three example scenarios where organizational IDs (***orgid***) are used to synchronize satellites:

Example 3.1. Import Content from Master to Slave Satellite

This example imports content from master to slave satellite:

```
satellite-sync --parent-sat=master.satellite.example.com -c channel-name
--orgid=2
```

Example 3.2. Import Content from an Exported Dump of an Organization

This example imports content from an exported dump of a specific organization:

```
$ satellite-sync -m /dump -c channel-name --orgid=2
```

Example 3.3. Import Content from Red Hat Network Hosted

This example imports content from Red Hat Network Hosted (assuming the system is registered and activated):

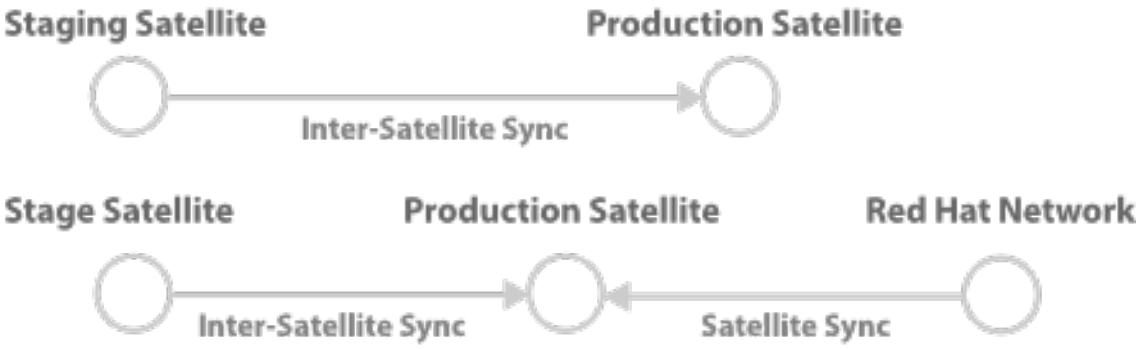
```
$ satellite-sync -c channel-name
```

3.3. ISS USE CASES

ISS can be used in several different ways, depending on the needs of your organization. This section provides examples of how you might choose to use ISS, and the methods for setting up and operating these cases.

Example 3.4. Staging Satellite

This example uses one satellite as a *staging satellite* to prepare content and perform quality assurance on the packages to ensure they are fit for production use. When content is approved to go to production, the production satellite can synchronize the content from the stage satellite.



1. Run the **satellite-sync** command to synchronize data with **rhn_parent** (usually Red Hat Network Hosted):

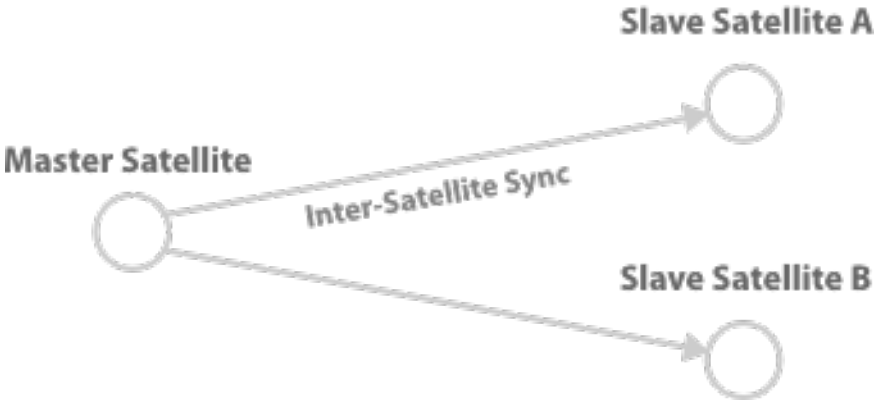
   ```
   satellite-sync -c your-channel
   ```

2. Run the following command to synchronize data from the staging server:

   ```
   satellite-sync --iss-parent=staging-satellite.example.com -c
   custom-channel
   ```

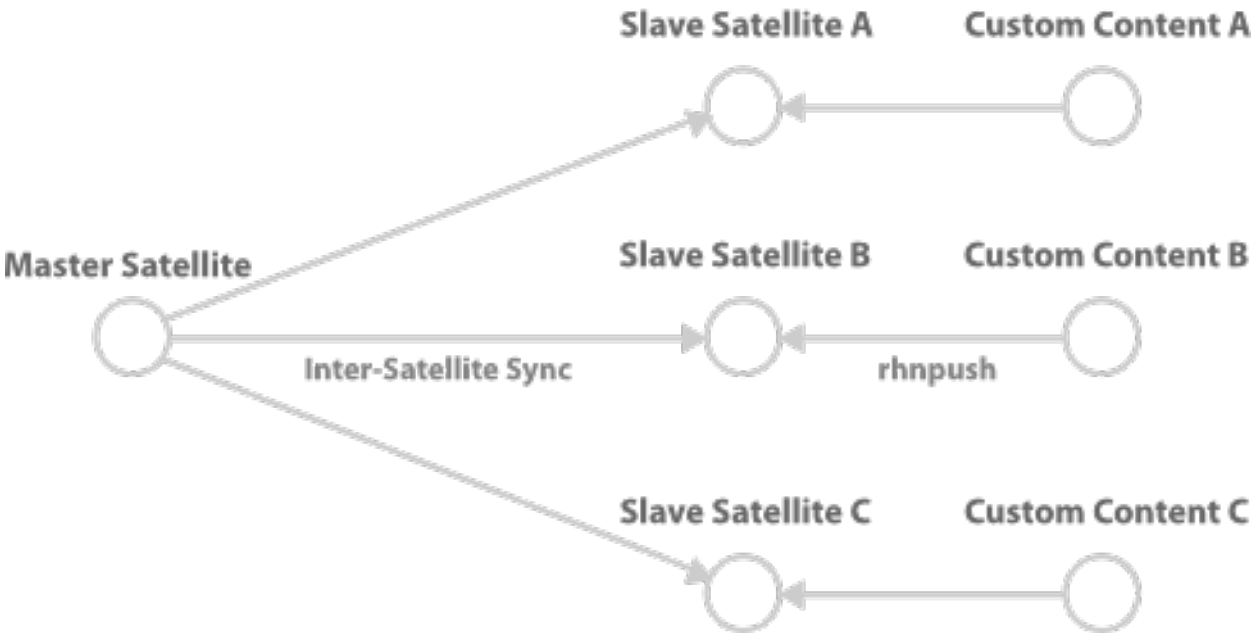
Example 3.5. Synchronized Slaves

In this example, the master satellite provides data directly to the slaves and changes are regularly synchronized.



Example 3.6. Slave Custom Content

This example uses the master satellite as a development channel, from which content is distributed to all production slave satellites. Some of the slave satellites have extra content that is not present in the master satellite channels. These packages are preserved, but all changes from master satellite are synchronized to the slaves.



Example 3.7. Bi-directional sync

In this environment, two RHN Satellite servers act as masters of each other and can synchronize content between them.

1. Ensure both satellites can share SSL certificates.
2. On the first satellite, open the `/etc/rhn/rhn.conf` file and set the `iss_parent` option to point to the hostname of the second satellite.
3. On the second satellite, open the `/etc/rhn/rhn.conf` file and set the `iss_parent` option to point to the hostname of the first satellite.

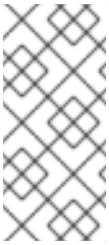
CHAPTER 4. ADVANCED COMMANDS

4.1. THE XML-RPC API

RHN Satellite 5.4.1 supports provisioning using the XML-RPC API.

The following API methods are used for kickstart profile and tree maintenance:

Table 4.1. XML-RPC Methods

XML-RPC Namespace	Usage
`kickstart`	Create, import, and delete kickstart profiles. Also used to list available kickstart trees and profiles.
`kickstart.tree`	Create, rename, update, and delete kickstart trees.
`kickstart.filepreservation`	List, create, and delete file preservation lists that can be associated to a kickstart profile. Once a file preservation list has been created, it can be associated to a kickstart profile by calling the `kickstart.profile.system.add_file_preservations` API method.
`kickstart.keys`	List, create, and delete cryptography keys (GPG/SSL) that can be associated to different kickstart profiles.  **NOTE** Once a cryptography key has been created, it can be associated to a kickstart profile by calling the `kickstart.profile.system.add_keys` API method.
`kickstart.profile`	Manipulate IP ranges, change the kickstart tree and the child channels, download the kickstart files associated with a profile, manipulate advanced options, manipulate custom options, and add pre- and post-scripts to a kickstart profile.
`kickstart.profile.keys`	List, add (associate), and remove (disassociate) activation keys associated to a kickstart profile.
`kickstart.profile.software`	Manipulate the list of packages associated to a kickstart profile.

XML-RPC Namespace	Usage
`kickstart.profile.system`	Manage file preservations, manage cryptography keys, enable/disable configuration management and remote commands, setup partitioning schemes, and setup locale information associated to a given kickstart profile.

The following API methods calls are used to re-provision a host and schedule guest installs:

- `system.provision_system`
- `system.provision_virtual_guest`

For more information on API calls refer to the API documentation available at `https://satellite.example.com/rpc/api`.

4.2. COBBLER

RHN Satellite uses Cobbler for provisioning. When the kickstart profiles, trees (distributions), and systems for provisioning are updated in RHN Satellite, they are synchronized to the Cobbler instance on the RHN Satellite host. This means that Cobbler can be used directly to manage provisioning.

The following table describes the Cobbler commands:

Table 4.2. Cobbler Commands

Command	Usage
`cobbler profile list`	Run this command on the RHN Satellite host to display a list of profiles
`cobbler distro list`	Display a list of kickstart trees, kernels, RAM disks, and other options
`cobbler system list`	Display a list of system records, created when a kickstart is scheduled
`cobbler profile report --name=profile-name` or `cobbler system report --name=system-name`	Display a more detailed output about a specific object
`cobbler profile edit --name=profile-name --virt-ram=1024`	Edit various parameters. This example will allocate each virtualized installation of a given profile 1GB of RAM.
`cobbler system edit --name=system-name --netboot-enabled=1`	Force a system to be reinstalled at the next reboot

Command	Usage
`cobbler system edit --name=system-name --profile=new-profile-name --netboot-enabled=1`	Assign a system to a new profile for reinstallation
`cobbler system find --profile=profile-name`	List all systems assigned to a profile
`cobbler system find --profile="abc" | xargs -n1 --replace cobbler system edit \--name={} --profile="def" --netboot-enabled=1`	Assign all systems currently set to the *abc* profile to the *def* profile and reinstall them the next time they reboot
`cobbler profile edit --name=profilename --kopts="variablename=3" --in-place`	Set an additional templating variable on a profile without modifying any of the other variables
`cobbler system edit --name=systemname --kopts="selinux=disabled asdf=jkl"`	Assign various variables to a system record, and disregard any old variables that might be set
`cobbler profile find --name="*webserver*" | xargs -n1 --replace cobbler profile edit --name={} --profile="RHEL5-i386"`	Set all new installations of any profile containing *webserver* as a string to use a profile named *RHEL5-i386*

Other Cobbler settings

There are only a few Cobbler settings that should be changed in `/etc/cobbler/settings` directly. The `pxe_just_once` option is one of these (described in Procedure 4.3, "Configuring Cobbler to use PXE"). The `server` option can also be changed to reflect the address or hostname of the RHN Satellite server.

After changing `/etc/cobbler/settings`, run the following command to pick up the changes:

```
/sbin/service cobblerd restart
cobbler sync
```

IMPORTANT

Do not adjust any other settings in `/etc/cobbler/settings`. RHN Satellite requires that this file remains in a certain configuration, determined by the RHN Satellite installer. Similarly, the `/etc/cobbler/modules.conf` file, which controls authentication sources, should remain as created by the RHN Satellite installer. Particularly, the authentication module must remain as `authn_spacewalk` and is not changeable.

Procedure 4.1. Configuring SELinux for use with Cobbler

SELinux support and a secure firewall is installed by default with Red Hat Enterprise Linux. To properly configure a Red Hat Enterprise Linux server to use Cobbler, SELinux must be configured to allow connections to and from the Cobbler server.

1. To enable SELinux for Cobbler support, set the SELinux Boolean to allow HTTPD web service components, using the following command:

   ```
   setsebool -P httpd_can_network_connect true
   ```

   The **-P** switch is essential, as it enables HTTPD connection persistently across all system reboots.

2. Set SELinux file context rules for TFTP to serve the boot image file, using the following command on the Cobbler server:

   ```
   semanage fcontext -a -t public_content_t "var/lib/tftpboot/.*"
   ```

3. IPTables must be configured to allow incoming and outgoing network traffic on the Cobbler server.

   If you have an existing firewall ruleset using iptables, add the following rules to open the Cobbler-related ports, as follows:

   For TFTP:

   ```
   /sbin/iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport
   69 -j ACCEPT
   /sbin/iptables -A INPUT -m state --state NEW -m udp -p udp --dport
   69 -j ACCEPT
   ```

   For HTTPD:

   ```
   /sbin/iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport
   80 -j ACCEPT
   /sbin/iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport
   443 -j ACCEPT
   ```

   For Cobbler:

   ```
   /sbin/iptables -A INPUT -m state --state NEW -m udp -p udp --dport
   25150 -j ACCEPT
   /sbin/iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport
   25150 -j ACCEPT
   ```

   For Koan:

   ```
   /sbin/iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport
   25151 -j ACCEPT
   ```

4. Save the firewall configuration:

   ```
   /sbin/iptables-save
   ```

5. Ensure that the configuration files are all synchronized by running the following command:

```
cobbler sync
```

6. Start the Satellite server:

```
/usr/sbin/rhn-satellite start
```

WARNING

Do not start or stop the **cobblerd** service independent of the Satellite service, as doing so may cause errors and other issues.

Always use **/usr/sbin/rhn-satellite** to start or stop RHN Satellite.

Procedure 4.2. Configuring Cobbler System Records

Cobbler system records are objects within Cobbler that keep track of a system and its associated kickstart profile. To perform PXE kickstarting a Satellite kickstart profile must be tied to the Cobbler system records for the machines you intend to kickstart.

1. Go to **System Details** → **Provisioning** for each system and select the kickstart profile to be associated.

2. Click **Create Cobbler System Record** to make the association.

3. This association will remain in place indefinitely unless you set the **pxe_just_once** option to true for any given machine. In that case the association will be broken after a successful kickstart. See Procedure 4.3, “Configuring Cobbler to use PXE” for more information about this setting.

Procedure 4.3. Configuring Cobbler to use PXE

Cobbler is set up to generate PXE configurations by default, but you might want to adjust the **pxe_just_once** configuration option to obtain the best possible PXE workflow in the BIOS.

1. Often, the BIOS order will be set to have the PXE boot occur first. This means that the system will not boot off the local disk unless the PXE server instructs it to do so remotely. This setup can create a *boot loop*, where the system continually reinstalls.

   In order to prevent boot loops, open the **/etc/cobbler/settings** file and add the following line:

```
pxe_just_once: 1
```

   This setting adds a ***$kickstart_done*** macro in the kickstart template, which tells the system to boot locally after it has completed the installation, instead of booting from the network.

2. If you include the ***pxe_just_once: 1*** setting, and you want to reinstall the system later on,

you will need to toggle the ***netboot-enabled*** flag on the system. This can be done using either the RHN Satellite web interface, or in Cobbler directly. When the system next reboots, it will perform a PXE install, and then return to booting from the local disk until the flag is reset.

If the BIOS is set to boot from local hard drives first, there is no need to have the ***pxe_just_once*** enabled. However, to reprovision the system using PXE, it will be necessary to zero out the MBR (master boot record).

Naming Conventions

To help keep data synchronized between RHN Satellite and Cobbler, RHN Satellite uses naming conventions for distributions and profiles. These naming conventions are important if you interact with Cobbler using the command line interface.

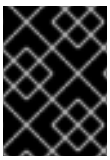
Distributions

`$tree_name:$org_id:$org_name` (if manually created)

`$tree_name` (if synchronized by RHN Satellite)

Profiles

`$profile_name:$org_id:$org_name`



IMPORTANT

Do not alter names that have been automatically generated by RHN Satellite. If the name is changed RHN Satellite can no longer maintain those items.

4.3. KOAN

The **koan** (kickstart over a network) utility allows RHN Satellite to be accessed remotely from hosts that have already been provisioned. Koan allows you to perform kickstart provisioning, create virtual guests (on virtual hosts), and can list the kickstarts available from the RHN Satellite host. It is available in the **koan** package.

Table 4.3. Koan Commands

Command	Usage
`man koan`	Read the **koan** manual page
`koan --replace-self --server=satellite.example.org --profile=profile-name` or **`koan --replace-self --server=satellite.example.org --system=system-name`**	Reprovision an existing system. Reboot after running this command to install the new operating system. This can also be used with upgrade kickstarts (for instance, to upgrade a large number of machines from one version of Red Hat Enterprise Linux to the next).

Command	Usage
koan --virt --server=satellite.example.org --profile=profile-name or **koan --virt --server=satellite.example.org --system=system-name**	Provision a virtual guest
koan --list=profiles --server=satellite.example.org or **koan --list=systems --server=satellite.example.org**	Query Cobbler to display a list of profiles or systems available for remote installation

CHAPTER 5. TROUBLESHOOTING

5.1. Web Interface

Q: I'm having problems with the RHN Satellite user interface. Which log files should I check?

A: If you experience errors viewing, scheduling, or working with kickstarts in the RHN Satellite user interface, check the **/var/log/tomcat5/catalina.out** log file.

For all other user interface errors, check the **/var/log/httpd/error_log** log file.

5.2. Anaconda

Q: I'm getting an error that says `Error downloading kickstart file`. What is the problem and how do I fix it?

A: This error is usually the result of a network issue. To locate the problem, run the **cobbler check** command, and read the output, which should look something like this:

```
# cobbler check
The following potential problems were detected:
#0: reposync is not installed, need for cobbler reposync,
install/upgrade yum-utils?
#1: yumdownloader is not installed, needed for cobbler repo add with -
-rpm-list parameter, install/upgrade yum-utils?
#2: The default password used by the sample templates for newly
installed machines (default_password_crypted in /etc/cobbler/settings)
is still set to 'cobbler' and should be changed
#3: fencing tools were not found, and are required to use the
(optional) power management features. install cman to use them
```

If **cobbler check** does not provide any answers, check the following:

Verify **httpd** is running

Verify **cobblerd** is running

Verify you can fetch the kickstart file using **wget** from a different host:

```
wget http://satellite.example.com/cblr/svc/op/ks/profile/rhel5-
i386-u3:1:Example-Org
```

Q: I'm getting a package installation error that says `The file chkconfig-1.3.30.1-2.i386.rpm cannot be opened.`. What is the problem and how do I fix it?

A: Clients will fetch content from RHN Satellite based on the ***--url*** parameter in the kickstart. For example:

```
url --url http://satellite.example.com/ks/dist/ks-rhel-i386-server-5-
u3
```

If you receive errors from Anaconda stating it can't find images or packages, check that the URL in the kickstart will generate a ***200 OK*** response. You can do this by attempting to **wget** the file located at that URL:

```
wget http://satellite.example.com/ks/dist/ks-rhel-i386-server-5-u3
--2011-08-19 15:06:55--  http://satellite.example.com/ks/dist/ks-rhel-
i386-server-5-u3
Resolving satellite.example.com... 10.10.77.131
Connecting to satellite.example.com|10.10.77.131|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: `ks-rhel-i386-server-5-u3.1'
2011-08-19 15:06:55 (0.00 B/s) - `ks-rhel-i386-server-5-u3.1' saved
[0/0]
```

If you get a response other than ***200 OK***, check the error logs to find out what the problem is. You can also check the actual file Anaconda tried to download by searching the **access_log** file:

```
# grep chkconfig /var/log/httpd/access_log
10.10.77.131 - - [19/Aug/2011:15:12:36 -0400] "GET
/rhn/common/DownloadFile.do?url=/ks/dist/ks-rhel-i386-server-
5-u3/Server  /chkconfig-1.3.30.1-2.i386.rpm HTTP/1.1" 206 24744 "-"
"urlgrabber/3.1.0 yum/3.2.19"
10.10.76.143 - - [19/Aug/2011:15:12:36 -0400] "GET /ks/dist/ks-rhel-
i386-server-5-u3/Server/chkconfig-
1.3.30.1-2.i386.rpm HTTP/1.1" 206 24744 "-" "urlgrabber/3.1.0
yum/3.2.19"
10.10.76.143 - - [19/Aug/2011:15:14:20 -0400] "GET /ks/dist/ks-rhel-
i386-server-5-u3/Server/chkconfig-
1.3.30.1-2.i386.rpm HTTP/1.1" 200 162580 "-" "urlgrabber/3.1.0
yum/3.2.19"
10.10.77.131 - - [19/Aug/2011:15:14:20 -0400] "GET
/rhn/common/DownloadFile.do?url=/ks/dist/ks-rhel-i386-server-
5-u3/Server/chkconfig-1.3.30.1-2.i386.rpm HTTP/1.1" 200 162580 "-"
"urlgrabber/3.1.0 yum/3.2.19"
```

If the requests are not appearing in the **access_log** file, the system might be having trouble with the networking setup. If the requests are appearing but are generating errors, check the error logs.

You can also try manually downloading the files to see if the package is available:

```
wget http://satellite.example.com/ks/dist/ks-rhel-i386-server-5-
u3/Server/chkconfig-1.3.30.1-2.i386.rpm
```

5.3. Cobbler and Koan

Q: Where are the Cobbler and Koan log files?

A: Cobbler logs data to the RHN Satellite logs, but also to **/var/log/cobbler/**.

Koan logs data to **/var/log/koan**.

5.4. Tracebacks

Q: I'm getting emails with "WEB TRACEBACK" in the subject. What should I do about them?

A: A typical traceback email might look something like this:

```
Subject: WEB TRACEBACK from satellite.example.com
Date: Wed, 19 Aug 2011 20:28:01 -0400
From: RHN Satellite <dev-null@redhat.com>
To: admin@example.com

java.lang.RuntimeException: XmlRpcException calling cobbler.
 at
com.redhat.rhn.manager.kickstart.cobbler.CobblerXMLRPCHelper.invokeMet
hod(CobblerXMLRPCHelper.java:72)
 at
com.redhat.rhn.taskomatic.task.CobblerSyncTask.execute(CobblerSyncTask
.java:76)
 at
com.redhat.rhn.taskomatic.task.SingleThreadedTestableTask.execute(Sing
leThreadedTestableTask.java:54)
 at org.quartz.core.JobRunShell.run(JobRunShell.java:203)
 at
org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleThreadPool.ja
va:520)
Caused by: redstone.xmlrpc.XmlRpcException: The response could not be
parsed.
 at redstone.xmlrpc.XmlRpcClient.handleResponse(XmlRpcClient.java:434)
 at redstone.xmlrpc.XmlRpcClient.endCall(XmlRpcClient.java:376)
 at redstone.xmlrpc.XmlRpcClient.invoke(XmlRpcClient.java:165)
 at
com.redhat.rhn.manager.kickstart.cobbler.CobblerXMLRPCHelper.invokeMet
hod(CobblerXMLRPCHelper.java:69)
 ... 4 more
Caused by: java.io.IOException: Server returned HTTP response code:
503 for URL: http://someserver.example.com:80/cobbler_api
 at
sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConn
ection.java:1236)
 at redstone.xmlrpc.XmlRpcClient.handleResponse(XmlRpcClient.java:420)
 ... 7 more
```

This indicates that there has been a problem with Cobbler communicating with the **taskomatic** service. Try checking the following:

Verify **httpd** is running

Verify **cobblerd** is running

Verify that there are no firewall rules that would prevent **localhost** connections

5.5. Registration

Q: The `rhnreg_ks` command is failing when I run it, saying `ERROR: unable to read system id`. What is the problem?

A: At the end of the kickstart file, there is a **%post** section that registers the machine to the RHN Satellite:

```
# begin Red Hat management server registration
mkdir -p /usr/share/rhn/
wget http://satellite.example.com/pub/RHN-ORG-TRUSTED-SSL-CERT -O
/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT
perl -npe 's/RHNS-CA-CERT/RHN-ORG-TRUSTED-SSL-CERT/g' -i
/etc/sysconfig/rhn/*
rhnreg_ks --serverUrl=https://satellite.example.com/XMLRPC --
sslCACert=/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT --activationkey=1-
c8d01e2f23c6bbaedd0f6507e9ac079d
# end Red Hat management server registration
```

In order, this will create a directory to house the custom SSL cert used by the RHN Satellite, fetch the SSL certificate to use during registration, search and replace the SSL certificate strings from the **rhn-register** configuration files, and then register to the RHN Satellite using the SSL certificate and an activation key. Every kickstart profile includes an activation key that assures that the system is assigned the correct base and child channels, and gets the correct system entitlements. If it is a reprovisioning of an existing system, the activation key will also ensure it is associated with the previous system profile.

If the **rhnreg_ks** command fails you might see errors like this in the **ks-post.log** log file:

```
ERROR: unable to read system id.
```

These errors will also occur if you attempt to perform an **rhn_check** and the system has not registered to the RHN Satellite.

The best way to troubleshoot this is to view the kickstart file and copy and paste the four steps directly at the command prompt after the kickstart has completed. This will produce error messages that are more detailed to help you locate the problem.

5.6. Kickstarts and Snippets

Q: What is the directory structure for kickstarts?

A: The base path where the kickstart files are stored is **/var/lib/rhn/kickstarts/**. Within this directory, raw kickstarts are in the **upload** subdirectory, and wizard-generated kickstarts are in the **wizard** subdirectory:

```
Raw Kickstarts: /var/lib/rhn/kickstarts/upload/$profile_name--
$org_id.cfg
Wizard Kickstarts: /var/lib/rhn/kickstarts/wizard/$profile_name--
$org_id.cfg
```

Q: What is the directory structure for Cobbler snippets?

A: Cobbler snippets are stored in **/var/lib/rhn/kickstarts/snippets**. Cobbler accesses snippets using the symbolic link **/var/lib/cobbler/snippets/spacewalk**.

```
Snippets:  /var/lib/rhn/kickstarts/snippets/$org_id/$snippet_name
```

IMPORTANT

RHN Satellite RPMs expect the Cobbler kickstart and snippet directories to be in their default locations, do not change them.

APPENDIX A. REVISION HISTORY

Revision 2-2.400 **2013-10-31** **Rüdiger Landmann**
Rebuild with publican 4.0.0

Revision 2-2 **Wed Jan 2 2013** **Athene Chan**
BZ#822406 updated information

Revision 2-1 **Wed Aug 22 2012** **Athene Chan**
BZ#822406 misspelled udp as ucp

Revision 2-0 **Tue Jul 17 2012** **Athene Chan**
BZ#822406 iptables command for cobbler corrected
BZ#702528 Kickstart Procedures need additional information

Revision 1-3 **Mon Aug 15 2011** **Lana Brindley**
Folded z-stream release into y-stream

Revision 1-2 **Wed Jun 15 2011** **Lana Brindley**
Prepared for publication

Revision 1-1 **Fri May 27 2011** **Lana Brindley**
Updates from translaters

Revision 1-0 **Fri May 6, 2011** **Lana Brindley**
Final QE Review edits
Prepared for translation

Revision 0-8 **Thu May 5, 2011** **Lana Brindley**
BZ#701822 - QE Review

Revision 0-7 **Thu April 14, 2011** **Lana Brindley**
Technical review feedback

Revision 0-6 **Wed March 23, 2011** **Lana Brindley**
Preparation for technical review

Revision 0-5 **Tue March 22, 2011** **Lana Brindley**
BZ#666435
BZ#666846
BZ#678080
BZ#682364

Revision 0-4 **Tue March 22, 2011** **Lana Brindley**
Troubleshooting

Revision 0-3 **Mon March 21, 2011** **Lana Brindley**
Multiple Satellites

Revision 0-2 **Thu March 17, 2011** **Lana Brindley**
Introduction
Kickstart
Advanced Commands
Some chapter restructuring

Revision 0-1 **Wed Jan 5, 2011** **Lana Brindley**

Completed new chapter structure

Revision 0-0 **Tue Dec 21, 2010** **Lana Brindley**

New document creation from original RHN Satellite Deployment Guide