Red Hat Network Satellite 5.3.0

Provisioning HowTo

Red Hat Network Satellite

Partha Aji
Michael DeHaan
Mike McCune
Dave Parker
Justin Sherrill

1801 Varsity Drive
Raleigh, NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588 Research Triangle Park, NC 27709 USA

Abstract

This document contains concise description and instructions for use of the new kickstart provisioning functionality in Red Hat Network Satellite 5.3.0. Until such time as proper formal documentation makes its way into Satellite itself, you can consider this to be authoritative.

All organizations need simple, yet powerful tools to deploy Red Hat Enterprise Linux systems. For many years, Red Hat Network Satellite has empowered companies to build repeatable, predictable and reliable deployment processes to ensure rapid repurposing of linux servers and desktops. Whether you have 10 systems or 10,000 systems, RHN Satellite can help you achieve this goal in a disciplined fashion. Now, after significant investment, RHN Satellite 5.3.0 has dramatically boosted the flexibility and power of its signature provisioning capabilities.

This document contains concise details and instructions for use of the new kickstart provisioning functionality in Red Hat Network Satellite 5.3.0. Until such time as proper formal documentation makes its way into Satellite itself, you can consider this to be authoritative.

1. Requirements

- A functional 5.3.0 Satellite in operation within your environment. If you are installing Satellite 5.3.0 on a new system, please refer to the instructions in the *Red Hat Network Satellite 5.3.0 Installation Guide* available at: *http://www.redhat.com/docs/manuals/satellite/Red_Hat_Network_Satellite-5.3.0/html/Installation_Guide*

- If you are upgrading from a previous version of Satellite consult the following Red Hat Knowledgebase article: How is Red Hat Network (RHN) Satellite upgraded to the most current version? (Article ID: 8610) *http://kbase.redhat.com/faq/FAQ_49_8610.shtm*

- To use the new provisioning functionality, you need one or more *target* machines — either physical, *bare metal* computer system(s) or virtual machine host(s). If you want to use Satellite's virtual machine provisioning functionality, your virtual machine host(s) should be configured with either the Xen or KVM virtualization technologies. Note that RHEL 5.4 and newer support KVM virtualization at this time.

2. Definitions and Terms

- Provisioning — The process of configuring a machine (real or virtual) to a predefined known state. Satellite ultimately accomplishes provisioning in all cases through the mechanism of kickstarting.
- * Kickstarting — A process of installing a Red Hat based system in an automated manner requiring little or no user intervention. Technically, *kickstart* refers to a mechanism in Red Hat's Anaconda installation program that allows you supply a concise description of the contents and configuration of a machine to Red Hat's installer, which it then acts on. Such a concise system definition is referred to in Satellite 5.3.0 as a Kickstart Profile.
- Kickstart Profile – A 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 5.3.0, 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 *Distro* - short for *distribution*.
- PXE or *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 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.

3. Provisioning Scenarios Supported

- New Installations — With Satellite 5.3.0 it is possible to provision systems that have not previously had any operating system installed (also known as *bare metal* installations).
- Virtual Installations — Satellite 5.3.0 supports KVM and Xen Fullvirt guests. Previously only Xen paravirt was supported as a virtualization type.
- Re-provisioning — Both physical and guest systems can be re-provisioned with Satellite 5.3.0, provided that they've been registered to a Satellite 5.3.0 instance (this was supported in Satellite 5.2 and earlier)

4. Overview of Preparing a Satellite for Provisioning

1. Sync content - refer to the *Satellite Installation Guide* for details
2. *Optional*: Manually setup a kickstart tree

3. Create a Kickstart Profile

4. Provision/reprovision machines

5. Kickstart Trees And Software Content

You must have at least one kickstart tree installed on your Satellite in order to use kickstart provisioning. Satellite supports both automatic and manual kickstart tree installation.

5.1. Automatically Installed Kickstart Trees

Automatic kickstart tree installation is a function of normal channel synchronization. For each distribution you intend to base kickstarts on, you must synchronize that distribution's base channel along with its corresponding RHN tools channel to your Satellite. If you are using a connected Satellite, you will synchronize your Satellite with Red Hat's servers directly. If your Satellite is disconnected, you'll need to obtain and sync from disconnected channel dumps (again available from Red Hat's servers). Regardless of the mechanism, the act of synching the channel automatically creates a corresponding kickstart tree for that distribution.

For example, if you want to use Red Hat Enterprise Linux 5 for x86 architecture, you would want to sync the **`rhel-i386-server-5`** channel and it's corresponding rhn-tools channel labeled **`rhn-tools-rhel-i386-server-5`**.

For more information on syncing content, refer to Section 6.2 of the *Satellite Installation Guide*.

5.2. Manually Installed Kickstart Trees

If you want to use Satellite to kickstart a custom distribution, a distribution not supported by Red Hat, or a beta version of Red Hat Enterprise Linux, you need to create a corresponding kickstart tree. To install a kickstart tree in one of these situations, you need to perform the following tasks:

1. Obtain the installation ISO from wherever is appropriate for the distribution

2. Copy the ISO to your satellite server and mount it to **`/mnt/iso`**

3. 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 distros. For example, you might copy a RHEL beta distribution's contents to **`/var/satellite/custom-distro/rhel-i386-server-5.3-beta/`**

4. Create a custom software channel with the Satellite web interface. (Navigate to **Channel** => **Manage Software Channels** => **Create Channel**) and create a base channel with an appropriate name and label. In keeping with the example RHEL beta version above, we might use the label **`rhel-5.3-beta`**.

5. Push the software packages (rpm files) from the tree location to the newly created software channel. Given our example above, you would do so by running

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

Note that the sub-directory within the tree may be different depending on your distribution. Once this step is complete, you may delete all of the RPM files from the appropriate directory within the tree path. In this example, run the following:

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

The packages are still stored on the Satellite server within the channel and thus are not needed within the kickstart tree. Although this entire step is optional, having the packages in a software channel allows them to be installed onto the system as needed (using **yum**) and not solely at the time of kickstart.

6. Create the Distribution within the Satellite's web interface. Navigate to **Systems** => **Kickstart** => **Distributions** => **Create New Distribution**. Provide an appropriate label, the full tree path (**/var/satellite/custom-distro/rhel-i386-server-5.3-beta/** in our case), select the base channel we created earlier, and then the correct Installer Generation. We would select **Red Hat Enterprise Linux 5** for this distribution. Finally select **Create Kickstart Distribution**.

7. You may want to clone a rhn-tools child channel from an existing Red Hat Enterprise Linux base channel to be a child of your newly created base channel.

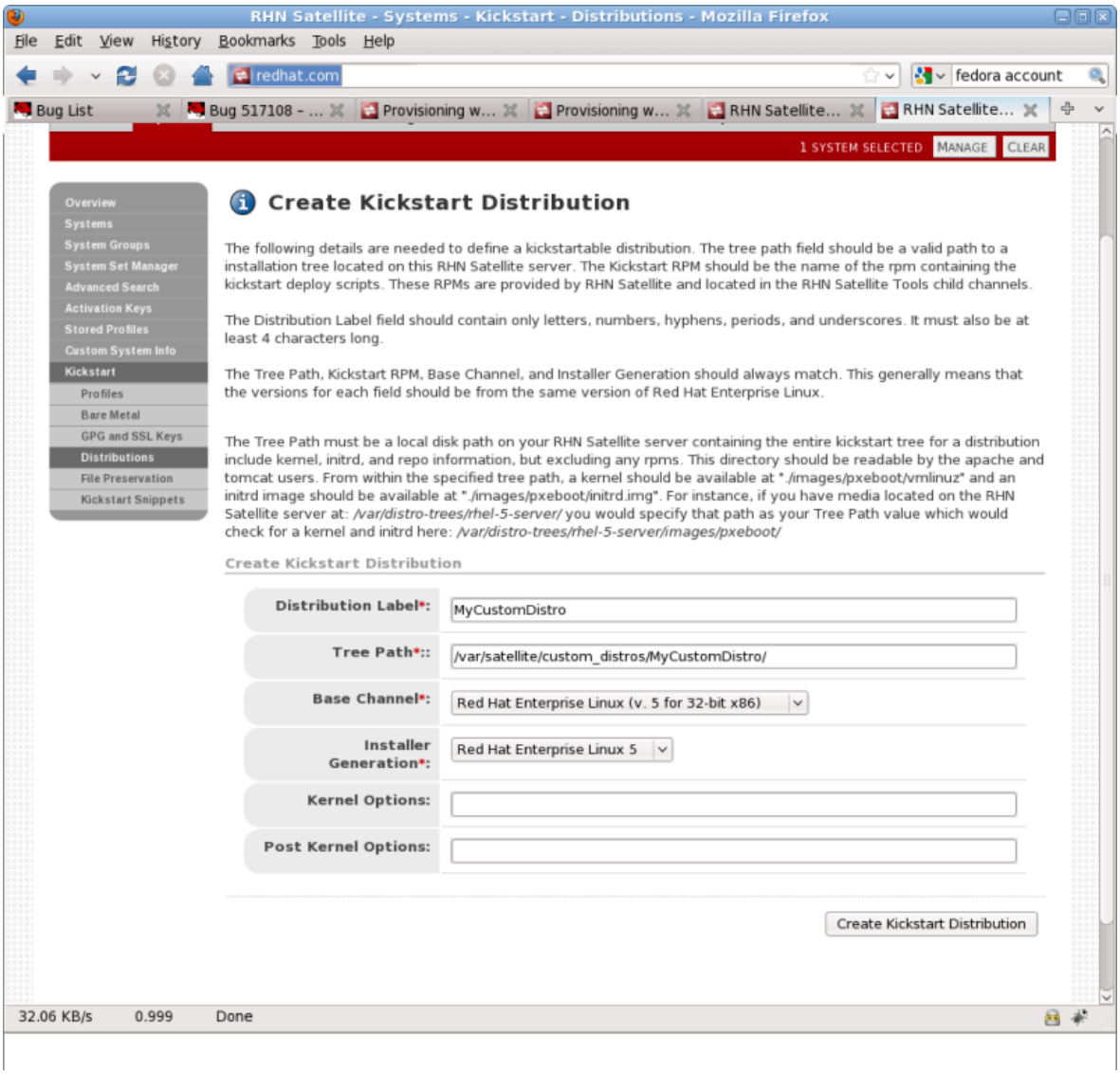

Figure 1. Creating Kickstart Distribution

5.3. Required distribution files

Satellite expects certain files to exist in specified locations within the Kickstart Tree and these locations will differ depending on the architecture of the system. The table below spells out where kernel and initrd are expected to reside for the different architectures.

arch	kernel	initrd image
s390x	<TREE_PATH>/images/kernel.img	<TREE_PATH>/images/initrd.img
PPC	<TREE_PATH>/ppc/ppc64/vmlinuz	<TREE_PATH>/images/pxeboot/vmlinux
All others	<TREE_PATH>/images/pxeboot/vmlinuz	<TREE_PATH>/images/pxeboot/initrd.img

Table 1. Required Distribution Files by Architecture

5.4. Required Packages

If using a custom distribution be sure that the packages **koan** and **spacewalk-koan** are available within a child channel of the distribution's base channel. These packages are available from any `rhn-tools` channel, and you may want to clone an existing rhn-tools channel in order to have access to these packages from your custom channel.

6. Kickstart Profiles

Kickstart profiles are the "recipes" that allow the installer to install the system with all of the configurations that the user wants. It is highly recommended that you review the "Kickstart Installations" Chapter of the *Red Hat Enterprise Linux Installation Guide* available at the following URL:

http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/ch-kickstart2.html

This guide discusses all of the options available for customizing the installation.

6.1. Virtualization Types

All Kickstart profiles have a virtualization type associated with them:

- None — This profile will be treated as not virtualized at all. Use for normal re-provisioning and bare metal installs.
- KVM Virtualized Guest — A KVM Guest. This is a supported feature for Red Hat Enterprise Linux 5.4 and newer.
- Xen Fully-Virtualized Guest — This option requires hardware support on the host, but does not require a modified operating system in the guest.
- Xen Para-Virtualized Guest — A virtual guest using Xen para-virtualiztion. Para-virtualiztion is the fastest virtualization mode and does not require hardware support, but does require a modified operating system. The following versions are supported as guests under para-virtualization:
  - Red Hat Enterprise Linux 5 (Any Update supported)
  - Red Hat Enterprise Linux 4 (Update 5 or later)
- * Xen Virtualization Host — A system that will host guests using Xen virtualization technology. This can support Xen paravirt guests, and can also support Xen fullvirt guests if the hardware itself supports it.

Note

Kickstarts created to be Xen hosts should include the **kernel-xen** package in the *%packages* section.

Kickstarts for KVM hosts should include the **qemu** package.

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

6.2. Creating Kickstart Profiles

RHN Satellite supports two distinct methods of kickstart profile creation: *Wizard-based* and *Raw*. Wizard style kickstart profiles are generated and maintained by Satellite logic, with many hooks for user modification of kickstart parameters. The Raw method is a mechanism by which you have complete control over the content of the kickstart file: you write the file completely yourself or upload an existing pre-made Kickstart file, and are entirely responsible for its contents.

6.2.1. Wizard Style Kickstarts

To create a wizard style Kickstart:

1. Click on **Systems** => **Kickstart** => **create a new kickstart profile**
2. Provide an appropriate **label**, select the desired **base channel** and **distribution**
3. Select the **Virtualization Type** desired
4. Select **next**
5. You will be presented with an option to use the default download location or use a custom one. Select the default unless you're using a custom distribution. (If you are using a custom distribution, enter the location of its tree via a URI (http and ftp are supported)
6. Click **next**
7. Enter the root password
8. Click **finish**

At this point, Satellite generates a fully functional Kickstart file.

If you would like to view the resulting file, click on **Kickstart File**.

6.2.2. Raw Style Kickstarts

To create a completely customizable Kickstart:

1. Click on **Systems** => **Kickstart** => **upload new kickstart file**
2. Provide an appropriate **label**
3. Select the desired **Distribution**
4. Select the appropriate virtualization type (see above)
5. If you have an existing kickstart file you can upload it using the file upload feature, otherwise simply copy and paste it into the **File Contents** box.

Note

Note that kickstarts uploaded here are actually templates, so you need to be aware of the following template escape sequences.

Since the raw kickstart is completely written by the user, the Satellite server does not handle using the specified distro as the **`url`** in the kickstart. Because of this, you will want to include your own `url --url` option. It should look similar to the following:

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

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

Here is a sample raw kickstart that you may want to use 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')
```

Please note that **`$http_server`** is used in place of the Satellite's domain name. This will be filled in when the kickstart template is rendered. Also the **`redhat_register`** snippet is used to handle registration.

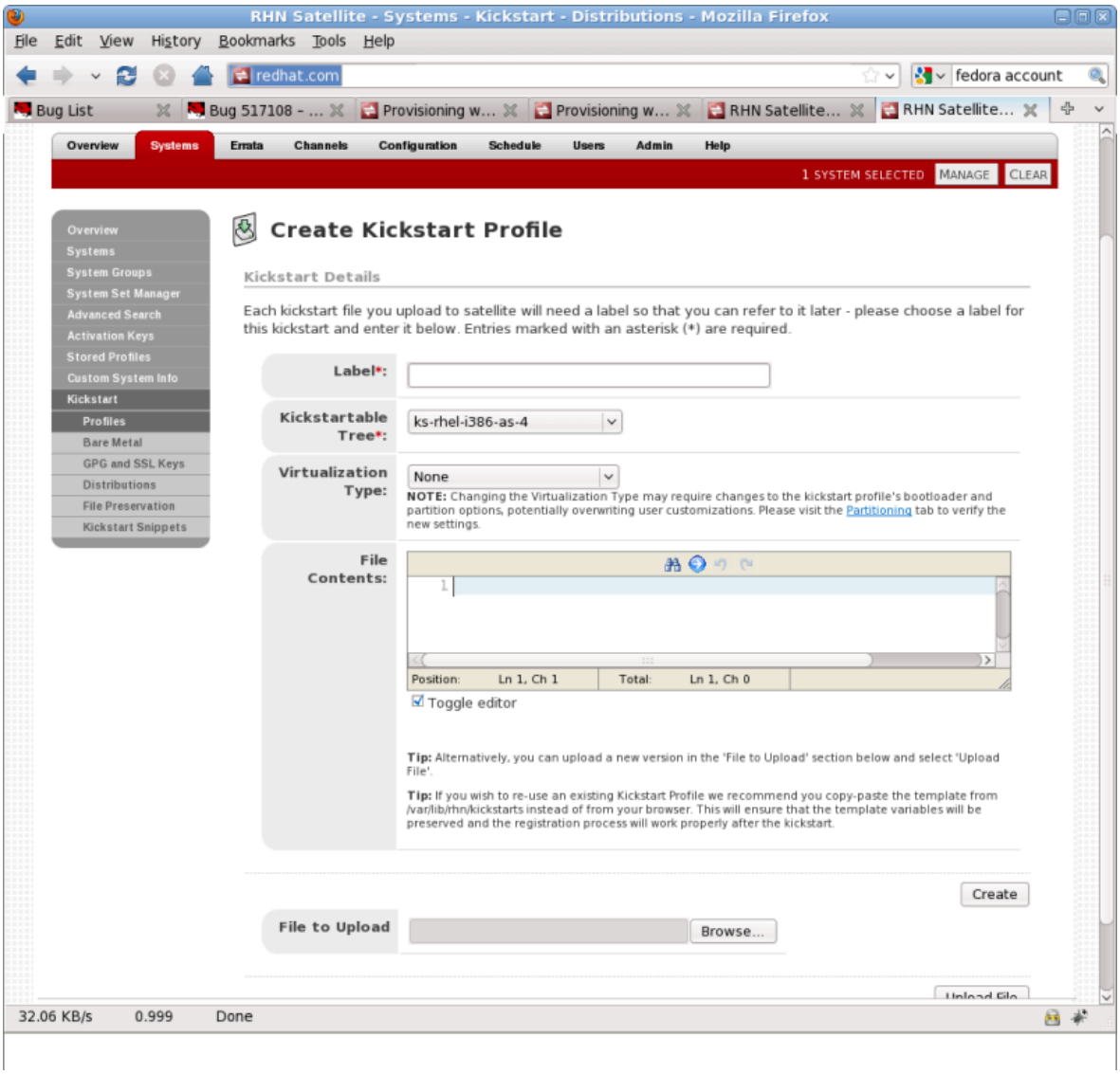

Figure 2. Raw Kickstart

7. Templating

One of the more powerful new features in Satellite 5.3.0 is Cheetah based kickstart templating. With this new capability, you can include variables, snippets (see below), and flow control statements such as for loops and if statements in your kickstart files.

7.1. Use Cases

There are a variety of reasons a user may want to use templating, such as:

- You might want to reuse a particular section of a kickstart, such as a disk partitioning section, between multiple kickstarts
- If you have, for example, multiple kinds of server roles such as DNS server, proxy server, and web server, all with their own package set. You could define a snippet for each role. For example web server might have the following snippet defined:

```
httpd
mod_ssl
mod_python
```

If you want to create a web server profile, include the web server snippet in the **%package** section of your Kickstart file. If you wanted a profile to be both a web server and a proxy server, you could include both snippets in the package section. Then if you wanted to add another package to the web server snippet, **mod_perl** for example, by updating the snippet all profiles that are using that snippet would be updated as well.

- You might want to perform certain actions in %post consistently across multiple kickstarts.

7.2. Variables

Templating allows for variables such as foo to be defined, and the value of those variables replaced wherever $foo is seen in 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 — the hierarchy is defined by Cobbler:

- Kickstart tree (distro in cobbler) parameters come first
- Kickstart Profile parameters override kickstart tree parameters
- System parameters override Profile parameters

If a variable is defined at the system level, it will override the same variable defined at the Profile or Distro levels. Likewise, if a variable is defined at the Profile level, it will override the same variable if defined at the kickstart tree (distro) level.

Note

Note that kickstart tree (distro) variables cannot be defined for non-custom (automatically generated) kickstart trees such as the ones you get when you do a satellite sync.

Refer to *https://fedorahosted.org/cobbler/wiki/KickstartTemplating* for more information.

7.3. Snippets

Snippets are similar to variables but can span many lines and can include variables in them. They can be included in a kickstart profile by using the text **$SNIPPET('snippet_name')**. You may make a snippet for a certain package list, one for a particular %post script, or for any text that would normally be included in a kickstart file.

The main purpose of snippets are to be able to reuse pieces of code between multiple kickstart templates and thus make each template easy to understand.

To manage snippets navigate to the **Systems** => **Kickstart** => **Kickstart Snippets** page. From here you can see Default Snippets that may not be edited, but may be used by any organization. These default snippets are provided to help make large tasks easier. For an explanation of common default

snippets see the **Default Snippet** section below. From this page you may also view Snippets created just for your organization on the **Custom Snippets** tab. You may also create a custom Snippet by clicking on the **create new snippet** link. Note, default snippets are stored on the Satellite server's file system in **`/var/lib/cobbler/snippets/`** while custom snippets are stored in the **`/var/lib/rhn/kickstarts/snippets/`** directory. Since Satellite stores its snippets for different orgs in different directories, any custom snippets will be used like the following:

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

The **1** in this case is the organization id. If you are not sure what text to insert in the kickstart in order to use your custom snippet, look for the **Snippet Macro** column on the snippet list, or on the snippet details page.

Snippets exist at a global level and do not share the same inheritance structure as variables. You may use variables within the snippets to change the way they behave depending on which system is requesting the kickstart.

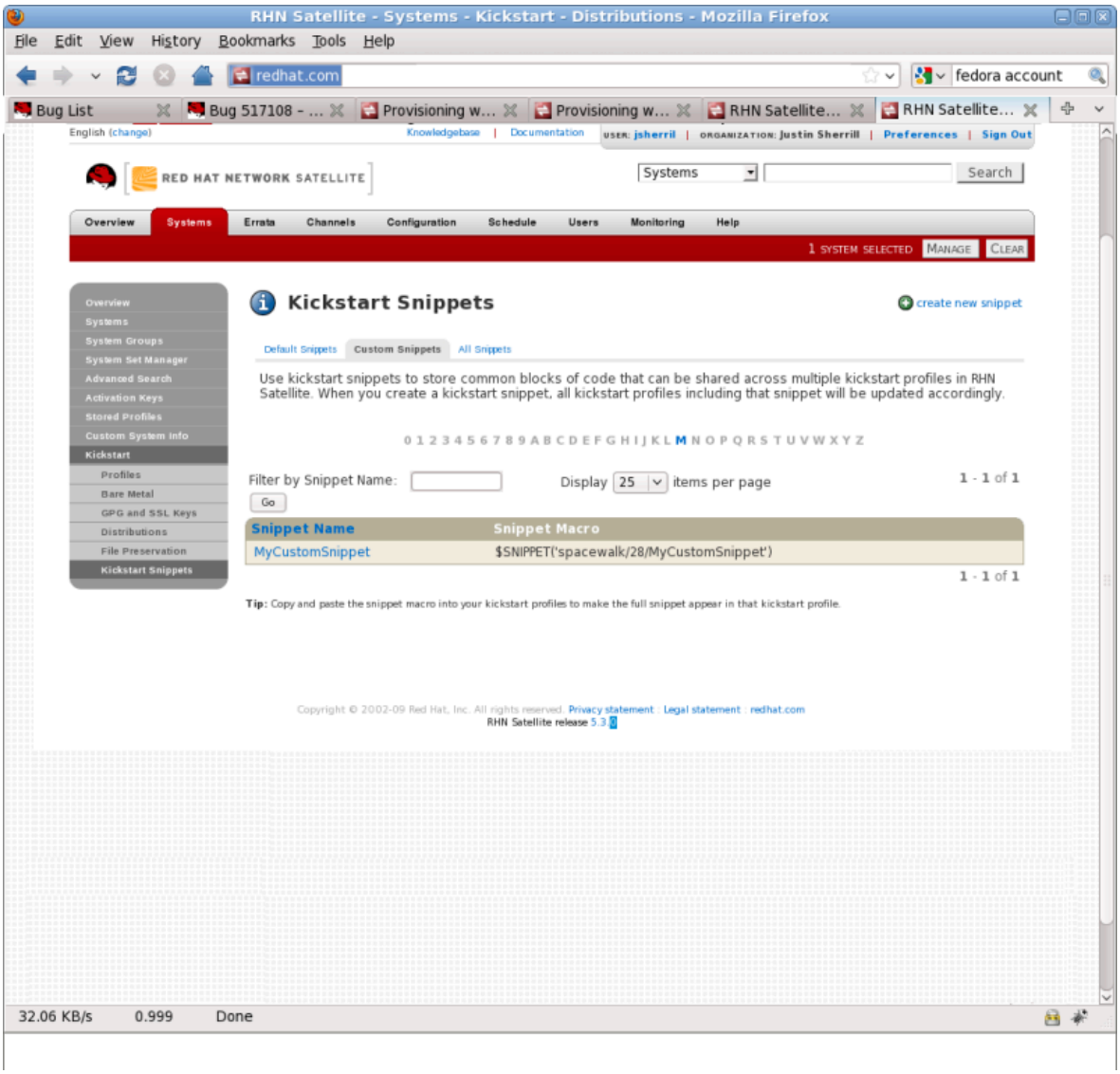

Figure 3. Kickstart Snippets

For more information, refer to *https://fedorahosted.org/cobbler/wiki/KickstartSnippets*.

7.3.1. Default Snippets

There are many snippets that ship by default and may be used in kickstarts written on or uploaded to the Satellite server. You may want to look at a template from a wizard style kickstart located in **`/var/lib/rhn/kickstarts/wizard/`** and see what default snippets are used and how they are used. One of the most useful ones is **`redhat_register`**.

The **`redhat_register`** snippet can be used to register machines to a Satellite server as part of the kickstart. It uses a special variable called **redhat_management_key** to register it to the server. Simply set that variable at either the system, profile, or distro level and then add **`$SNIPPET('redhat_register')`** to a %post section of your kickstart. Any wizard style kickstarts that are generated by the Satellite server will already include this snippet in it's pre-made `%post` section.

7.3.2. Escaping Special Characeters

Since the **$** and **#** characters are used during templating for specifying variables and control flow, you should not use these characters within scripts without escaping them.

So for example, if you were writing a bash script in a `%post` section:

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

The templating engine would try to find a variable named `$foo` and would fail if **`foo`** did not exist as a variable. There are a few ways to escape the **$** symbol so it shows up as a bash variable. The simplest is with a backslash:

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

`\$foo` will be rendered as **`$foo`** within the kickstart.

A second method is to wrap the entire script in **`#raw ... #endraw`** :

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

All `%pre` and `%post` scripts created using the wizard style kickstarts are wrapped with **`#raw...#endraw`** by default. This can be toggled using the **Template** checkbox available when editing a `%post` or `%pre` script.

The final method is simply by including **`#errorCatcher Echo`** in the first line of your kickstart. 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, but you may want to include it in the raw kickstarts you create yourself.

If you would like more information about Cheetah and the constructs that can be used for writing kickstart templates, the Cheetah User's Guide should be very helpful:

http://www.cheetahtemplate.org/docs/users_guide_html/

8. Kickstarting a Machine

8.1. Bare Metal

Satellite provides three mechanisms by which you can provision *bare metal* machines — machines that have no operating system or that have the wrong operating system installed:

1. Boot Anaconda-style operating system installation disk
2. PXE boot
3. Boot Cobbler boot disk

8.1.1. Booting from an Anaconda Style Installation Disk

Simply boot the selected system using an installation disc that matches your kickstart. For example, if your kickstart was configured to use the **`ks-rhel-i386-server-5-u2`** kickstart tree, you must boot with the Red Hat Enterprise Linux 5.2 i386 installation disc. When the boot prompt comes up, simply type:

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

The system will boot, download the kickstart, and re-install itself.

8.1.2. PXE Booting

PXE booting is a very convenient method of installing and reinstalling your physical systems, but does come with a few requirements:

- You must have a DHCP server, even if your systems are to be configured statically after installation.
- As DHCP does not normally cross network (router) boundaries, you will need to make special provision to ensure that all of your machines can connect to your dhcp server(s) in the event your machines reside on multiple networks. Options here include multi-homing your DHCP server (either real or trunked vlan) and configuring your routers or switches to pass DHCP across network boundaries.
- You must be able to configure your DHCP server to point to the PXE server (the Satellite server), by setting the *`next-server`* address for the systems you want to be managed by Satellite.
- Each system you have must support PXE booting at the BIOS level. Nearly all recent hardware should be able to do this.

8.1.2.1. Configuring an External DHCP Server

To configure your DHCP server (assuming you are using ISC DHCPd) to point to the PXE server, simply add the following to your configuration in **`/etc/dhcpd.conf`**:

```
next-server satellite.example.com;
filename "pxelinux.0";
```

Replace *`satellite.example.com`* with your Satellite Server's FQDN. After restarting your dhcp server, any clients that attempt to PXE boot will try to use the Satellite server as its PXE server.

You may want to assign the next-server only for a particular subnet or set of computers. Refer to the documentation on your DHCP server for information.

8.1.2.2. Enabling the TFTP server

As part of the Satellite installation itself, the TFTP server should be enabled. You can check this by running:

```
chkconfig --list tftp
```

You can enable it by running:

```
chkconfig tftp on
service xinetd restart
```

8.1.2.3. Cobbler configuration

Cobbler is already set up to generate pxe configurations, but you may want to adjust the `pxe_just_once` configuration option depending on how your machines BIOSes are configured, for the best possible PXE workflow.

A common setup has PXE occur first in the BIOS order, effectively *not* booting off the local disk unless the PXE server instructs the system to do so remotely. By having `pxe_just_once: 1` (enabled) in **`/etc/cobbler/settings`**, it will prevent "boot loops" where the system continually reinstalls. What happens is that the `$kickstart_done` macro in the kickstart templates will expand into a directive that indicates to the cobbler server that the system will then boot locally, instead of booting from the network. Then, to reinstall the system, the `netboot-enabled` flag on the system can be toggled back on via the Satellite GUI or Cobbler. Once enabled, the next time the system power cycles it will PXE install instead of booting locally. At the end of each install, the server will trip the netboot-enabled flag back to **`off`** again to tell the system to boot to the local hard drive the next time it powers up. Note that if your kickstart is missing the `$kickstart_done` line in `%post`, this will not work, and boot loops will occur.

With `pxe_just_once` set to **`0`**, the netboot enabled flag will *not* be disabled after an install, so if PXE is first in your BIOS boot order, the system will loop indefinitely. If you have the BIOS of the system set up to boot to local hard drives first, though, there is no need to set `pxe_just_once` enabled, but to re-PXE a system it is then neccessary to zero out the MBR of that system.

8.1.2.4. Cobbler System Record

Cobbler system records are objects within cobbler that keep track of a system and its associated kickstart profile. To do PXE kickstarting you'll need to ensure that a Satellite kickstart profile is tied to Cobbler system records corresponding to the machines you intend to PXE kickstart to that profile. To make this association:

1. Visit the System details page of each system in question and click on the **Provisioning** link
2. Select the kickstart profile you want to associate it with
3. Click the **Create Cobbler System Record** button.

Once you've made this association, it will remain in place forever unless you have set `pxe_just_once` to true in cobbler for any given machine. In that case the association will be broken after a successful kickstart.

Without this association, a machine that PXE bootstraps to a Satellite server will be presented with a menu of kickstart profiles which requires manual interaction.

8.1.2.5. Cobbler Boot ISO

The Cobbler boot iso is a disk image that can be built on your Satellite server and burned to a cd or dvd. You can then boot any system with it. When you do you will see a menu of available kickstarts similar to the one you would see if you PXE boot a machine off a Cobbler server without a system record. Simply select the kickstart you want, and the system will start to install itself. Any time you add a kickstart within Satellite, you will need to recreate the ISO and re-burn it to an optical disc.

To create a boot iso, log in to your Spacewalk server as root and run **`cobbler buildiso`**. The ISO will contain all kernel/initrd images stored in your Satellite along with all associated kernel argument settings. Kickstart files will be sourced remotely. This means that changes to the kickstart templates can be made without having to re-burn the CD. If you create a new kickstart profile and want to use it via the cobbler boot iso, you will need to recreate a fresh disc.

Note

Due to issues with the version of syslinux shipped with Red Hat Enterprise Linux 4, this command will not work unless the Satellite is running on Red Hat Enterprise Linux 5. Also since syslinux is not available for s390x, it is not possible to use this on a satellite running on s390x.

8.2. Re-Provisioning

Re-provisioning is the act of reinstalling an existing system. It could be reinstalled to the same version and release, or to a completely new version. When you re-provision through the Satellite web interface your system will use the same system profile that it had before it was re-provisioned. This can be useful as much of the information and settings about the system such as its history will be preserved.

You can schedule a re-provision from the **provisioning** tab while viewing a system. If you would like to configure additional options click on the **Advanced Options** page. On this page you can configure details such as kernel options, networking information, and package profile synchronizations. The **Kernel Options** section pertains to kernel options used during kickstart. **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.

For example:

- If you want to open up a vnc connection so you can 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

Note that this requires a system that is accessible over your network and already registered to Satellite. If you are reinstalling a system that is not registered to Satellite, there are several options:

- PXE
- Use **`cobbler buildiso`**
- Install koan on the system and use the koan command line tool, pointing at the Satellite server, to install a named profile

Koan is covered in a later section.

8.2.1. File Preservation

If you would like to keep some files across a re-provision you can use Satellite's *File Preservation* mechanism. This mechanism stores files temporarily during the kickstart and restores them at the end. To create a file preservation list:

- Go to **Systems** => **Kickstart** => **File Preservation Lists** and create a list of files to preserve
- After creation, associate your list with a kickstart:
  - Go to **Systems** => **Kickstart** => **Profiles**
  - Select on the desired profile
  - Select **System Details** => **File Preservation**
  - Select your file preservation list

Note

File preservation lists are only available on Wizard-style kickstarts and are only available during re-provisioning.

8.3. Virtualized Guest Provisioning

The following forms of Virtual Guest Provisioning is supported in Satellite 5.3:

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

Virtualization Type is set when when creating your kickstart profile. To provision a guest regardless of its type, follow the following steps:

1. Ensure the host system has a **Virutalization** or **Virtualization Platform** entitlement.
2. Go to the Guest Provisioning page at **Systems** => click on the desired virtual host => **Virtualization** => **Provisioning**
3. Select the kickstart profile you would like and enter a guest name
4. Select **Schedule Kickstart and Finish**.

If you would like to configure additional parameters such as guest memory and cpu usage, simply click on the **Advanced Configuration** button. The following can be configured:

- Network (static/dhcp)
- Kernel Options
- Package profile sync (When the kickstart finishes the system will sync its package profile to that of another system or stored profile)
- Memory Allocation (RAM, Default of 512)
- Virtual Disk Size
- Virtual Cpus (Default of 1)
- Virtual Bridge (The networking bridge used for the install. **`xenbr0`** is the default for Xen provisioning and **`virbr0`** is the default for KVM. Note that **`virbr0`** is not an actual bridge, so in that case it is best to configure host networking to create an actual bridge if outside networking is desired — and it almost always is — **`xenbr0`** is an actual bridge, and usage is recommended if it exists).
- Virtual Storage Path (Path to either a file, LVM Volume Group, directory, or a 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`**)

8.4. Provisioning Through an RHN Proxy

If you have an RHN Proxy installed and registered to your satellite you can easily provision through it. When provisioning a virtual guest or doing a re-provisioning of a system, simply select the desired proxy from the 'Select Satellite Proxy' drop down box. If you are doing a bare metal install, you can simply replace the Satellite's FQDN with that of the proxy's. For example if the url to your kickstart file is:

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

Then to kickstart through the proxy, use:

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

9. Advanced Topics

9.1. API

Red Hat Satellite 5.3.0 supports provisioning functionality using the XMLRPC API. The API supports everything from scheduling re-provisioning to modifying kickstart trees or profile details.

These methods facilitate kickstart profile and tree maintenance:

XMLRPC Namespace	Usage
kickstart	create, import, and delete kickstart profiles. Also to list available kickstart trees and profiles.
kickstart.tree	create, rename, update and delete kickstart trees.
kickstart.filepreservation	list, create, delete file preservation lists that can be associated to a kickstart profile. Note: once a file preservation list has been created, it can be associated to a kickstart profile by calling kickstart.profile.system.add_file_preservations.
kickstart.keys	list, create, 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 channel, download kickstart file associated to a profile, manipulate advanced options, manipulate custom options, and add pre/post scripts associated 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.
kickstart.profile.system	manage file preservations, manage cryptography keys, enable/disable config management and remote commands, setup partitioning schemes, setup locale information associated to a given kickstart profile.

Table 2. XMLRPC Methods

Additionally, the following API methods calls may be used to re-provision host and schedule guest installs.

- system.provision_system
- system.provision_virtual_guest

For more information on these API calls and others refer to the api documentation available on https://*`sat FQDN`*/rhn/rpc/api replacing *`sat FQDN`* with your Satellite server.

9.2. Cobbler On the Command Line

Satellite uses Cobbler to facilitate provisioning. When the kickstart profiles, trees (distributions) and systems for provisioning are updated in satellite, they are synced to the Cobbler instance on the satellite host. This means that you can use cobbler directly to manage their provisioning if you prefer.

To get a list of profiles run the following command in a terminal on host where the satellite is installed:

```
sudo cobbler profile list
```

To get a list of kicktstart trees (and kernels, initrds, and other options) run:

```
sudo cobbler distro list
```

To get a list of system records (which are created when a kickstart is scheduled) run:

```
sudo cobbler system list
```

To show more detailed output about a specific object, use the "report" command:

```
sudo cobbler profile report --name=profile-name
sudo cobbler system report --name=system-name
```

Various parameters can be tweaked just as with the Satellite GUI, for instance, asking that each virtualized install of a given profile get 1 GB of RAM:

```
sudo cobbler profile edit --name=profile-name --virt-ram=1024
```

9.3. Cobbler Command Line: Next Steps

Setting a system (see `pxe_just_once` above) to be reinstalled at next reboot

```
sudo cobbler system edit --name=system-name --netboot-enabled=1
```

Assigning a system to a new profile for reinstallation.

```
sudo cobbler system edit --name=system-name --profile=new-profile-name --
netboot-enabled=1
```

Listing all systems assigned to a particular profile

```
sudo cobbler system find --profile=profile-name
```

Assigning all systems currently set to the "abc" profile to the "def" profile and reinstalling them the next time they power cycle.

```
sudo cobbler system find --profile="abc" | xargs -n1 --replace cobbler
 system edit --name={} --profile="def" --netboot-enabled=1
```

Setting an additional templating variable on a profile without modifying any of the other variables

```
sudo cobbler profile edit --name=profilename --kopts="variablename=3" --in-
place
```

Assigning various variables to a system record, disregarding old variables that might be set

```
sudo cobbler system edit --name=systemname --kopts="selinux=disabled
 asdf=jkl"
```

Setting all new installs of any profile containing `webserver` as a string to use a profile named **`RHEL5-i386`** instead of RHEL 4 for any new installs.

```
sudo cobbler profile find --name="*webserver*" | xargs -n1 --replace
 cobbler profile edit --name={} --profile="RHEL5-i386"
```

Generating a net install ISO to install systems that cannot PXE

```
sudo cobbler buildiso [--help]
```

9.4. Naming Conventions

Satellite manipulates Cobbler distributions, profiles, and systems. To help keep data in sync between itself and Cobbler, Satellite relies on some naming conventions for these object types:

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

  Or `$tree_name` (if synced by Satellite Sync)

- profiles: `$profile_name:$org_id:$org_name`
- systems: FIXME

You will encounter these somewhat cryptic looking names if you choose to interact with Cobbler directly at the command line. Note that it is vitally important that you *not* alter Satellite generated names so long as you want to allow Satellite to maintain the objects in question.

Note

Satellite does not create or recognize Cobbler "repo" objects. Satellite's equivalent derives from its notion of channels and is a function of a layer of logic over them. It takes the form of a special URL which Cobbler is made to use instead.

9.5. Other Cobbler settings

There are only a few settings that should concern Satellite users. `pxe_just_once` is mentioned earlier in the PXE section. `server:` should be set to the address or hostname of the Satellite server.

No other settings should be tweaked in **`/etc/cobbler/settings`** as Satellite assumes them to be in a certain configuration. The settings file itself is layed down by the Satellite installer.

Similarly, **`/etc/cobbler/modules.conf`**, which controls authentication sources, should remain as installed by the Satellite installer. (The authentication module choice must remain `authn_spacewalk` and is not changeable).

After changing **`/etc/cobbler/settings`** (such as the `server` parameter or `pxe_just_once`) it is important to run the following so that the settings take effect:

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

9.6. Using Koan directly

koan (kickstart over a network) is a client utility that lets you invoke Satellite's (and Cobbler's) functionality remotely from already provisioned hosts. With it you can exercise kickstart provisioning, create virtual guests (on VM hosts), and list the kickstarts available from the Satellite host. It is available in the **koan** package.

You can read the **koan** manpage by running:

```
man koan
```

You can re-provision an existing system using koan by using one of the following methods:

```
koan --replace-self --server=satellite.example.org --profile=profile-name
```

Or:

```
koan --replace-self --server=satellite.example.org --system=system-name
```

Reboot after running the above command to install the new OS. This can also be used with upgrade kickstarts if desired (for instance, to upgrade a large number of machines between RHEL 4 and RHEL 5)

You can provision a virtual guest by using one of the following methods:

```
koan --virt --server=satellite.example.org --profile=profile-name
```

Or:

```
koan --virt --server=satellite.example.org --system=system-name
```

You can query **cobbler** to see what is available to install remotely by using one of the following methods:

```
koan --list=profiles --server=satellite.example.org
```

Or:

```
koan --list=systems --server=satellite.example.org
```

10. Troubleshooting

10.1. Web Interface errors

/var/log/tomcat5/catalina.out — Check this logfile first if you get errors in the RHN Satellite WebUI when viewing, scheduling or working with kickstarts.

/var/log/httpd/error_log — Check this logfile second for possible sources of WebUI errors

10.2. Anaconda Startup errors

If you get errors during the initiation of Anaconda where it can't find the kickstart file:

```
      +-------------+ Error downloading kickstart file +-------------+

      |                                                               |

      | Unable to download the kickstart file.  Please modify the     |

      | kickstart parameter below or press Cancel to proceed as an    |

      | interactive installation.                                     |

      |                                                               |

      | dhat.com/cblr/svc/op/ks/profile/rhel5-i386-u3:1:Example-Org_  |

      |                                                               |

      |            +----+                      +--------+            |

      |            | OK |                      | Cancel |            |

      |            +----+                      +--------+            |
```

```
       |                                                                 |

       |                                                                 |

       +-----------------------------------------------------------------+
```

You can check the following items:

1. Verify **`httpd`** is running on your RHN Satellite
2. Verify **`cobblerd`** is running
3. Verify you can fetch the above file using wget from a different host. For example:

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

4. run **`cobbler check`** from the CLI. You should see only this output:

   ```
   # 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 you see complaints about problems with **`httpd`**, **`cobblerd`**, or others, you must resolve those issues.

Note

In the case of a system reprovision, check the following URL:

http://*`sat_fqdn`*/cblr/svc/op/ks/system/$system_name:$org_id

In the case of a guest reprovision you can optionally check the following URL:

http://*`sat_fqdn`*/cblr/svc/op/ks/system/$system_name:$org_id:$guest_name

10.3. Anaconda content errors

```
        +------------------+ Package Installation +------------------+
        |                                                             |
        +-------------------------+ Error +-------------------------+
        |                                                             |
        | The file chkconfig-1.3.30.1-2.i386.rpm cannot be opened.    |
        | This is due to a missing file, a corrupt package or         |
        | corrupt media.  Please verify your installation source.     |
        |                                                             |
        | If you exit, your system will be left in an inconsistent    |
        | state that will likely require reinstallation.              |
        |                                                             |
        |                                                             |
        |          +--------+                   +-------+           |
        |          | Reboot |                   | Retry |           |
        |          +--------+                   +-------+           |
        |                                                             |
        |                                                             |
        +-------------------------------------------------------------+
```

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

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

If you receive errors from Anaconda stating it can't find images or packages you should first check that the above URL will generate a 200 response:

```
 wget http://somehost.example.com/ks/dist/ks-rhel-i386-server-5-u3
 --2009-08-19 15:06:55--  http://somehost.example.com/ks/dist/ks-rhel-i386-
server-5-u3
 Resolving somehost.example.com... 10.10.77.131
 Connecting to somehost.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'
 2009-08-19 15:06:55 (0.00 B/s) - `ks-rhel-i386-server-5-u3.1' saved [0/0]
```

If you don't get a 200 response check the error logs. After checking the base URL you can check the actual file Anaconda tried to download:

```
 # grep chkconfig /var/log/httpd/access_log
 10.10.77.131 - - [19/Aug/2009:15:12:36 -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/2009:15:12:36 -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/2009:15:14:20 -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/2009:15:14:20 -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 those requests are not appearing in the **`access_log`** file, the system may be having trouble with the networking setup.

If those requests are appearing but are generating errors, see the previously mentioned log files for errors.

You can also try manually downloading the files:

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

Then you can see if the package is available.

10.4. Cobbler log files

In addition to Satellite logs, cobbler also keeps some data in **`/var/log/cobbler/`**. When troubleshooting failed virtual installs, **`koan`** also saves the **`libvirt`** guest creation XML in **`/var/log/koan`**, which can occasionally be useful.

10.5. Tracebacks from Taskomatic

If you receive emails such as:

```
Subject: WEB TRACEBACK from someserver.example.com
Date: Wed, 19 Aug 2009 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.invokeMethod(CobblerXML
 at
 com.redhat.rhn.taskomatic.task.CobblerSyncTask.execute(CobblerSyncTask.java:76)
 at
 com.redhat.rhn.taskomatic.task.SingleThreadedTestableTask.execute(SingleThreadedTest
 at org.quartz.core.JobRunShell.run(JobRunShell.java:203)
 at org.quartz.simpl.SimpleThreadPool
$WorkerThread.run(SimpleThreadPool.java: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.invokeMethod(CobblerXML
 ... 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(HttpURLConnection.java:12
 at redstone.xmlrpc.XmlRpcClient.handleResponse(XmlRpcClient.java:420)
 ... 7 more
```

This indicates there is a problem found between the 'taskomatic' service and 'cobblerd' communicating. Check:

1. Verify httpd is running on your RHN Satellite
2. Verify cobblerd is running
3. Verify no firewall rules that would prevent localhost connections from one process to the above path.

10.6. Registration Issues

At the end of the kickstart there is a %post section that will register your kickstarted machine to the RHN Satellite:

```
 # begin Red Hat management server registration
 mkdir -p /usr/share/rhn/
 wget http://someserver.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://someserver.example.com/XMLRPC --sslCACert=/
usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT
   --activationkey=1-c8d01e2f23c6bbaedd0f6507e9ac079d
 # end Red Hat management server registration
```

Breaking this down into the 4 steps you have:

```
1) mkdir -p /usr/share/rhn/
```

Creating a directory to house the custom SSL cert used by the RHN Satellite

```
2) wget http://someserver.example.com/pub/RHN-ORG-TRUSTED-SSL-CERT -O /usr/
share/rhn/RHN-ORG-TRUSTED-SSL-CERT
```

Fetch the SSL cert to use during registration

```
3) perl -npe 's/RHNS-CA-CERT/RHN-ORG-TRUSTED-SSL-CERT/g' -i /etc/sysconfig/
rhn/*
```

Search/replace the SSL cert strings from the **`rhn-register`** configuration files.

```
4) rhnreg_ks --serverUrl=https://someserver.example.com/XMLRPC --
sslCACert=/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT --activationkey=1-
c8d01e2f23c6bbaedd0f6507e9ac079d
```

Register to the RHN Satellite with the SSL cert and an activation key. Every Kickstart Profile includes an Activation Key that assures that the system is assigned the correct base channel, gets the proper System Entitlements and is associated with the previous System Profile if you are re-provisioning an existing system.

If the **`rhnreg_ks`** command fails you may see errors in the ks-post.log indicating:

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

And calls to **`rhn_check`** return the error above you know the system failed to register to the RHN Satellite.

The best way to troubleshoot this is to view the kickstart file and copy-paste the 4 steps from above into the CLI and run them after the system comes back from kickstarting. Generally **`rhnreg_ks`** will produce usable error messages which should help you figure out what is failing during registration.

10.7. Directory structure for Kickstarts and Snippets

- Kickstarts — The base path where the kickstart files are stored is **`/var/lib/rhn/kickstarts/`**. Within this directory, raw (non-wizard generated) kickstarts reside in the subdirectory **`upload`** while wizard generated ones are in the **`wizard`** subdirectory, thus:

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

- Snippets — Cobbler Snippets are stored in **`/var/lib/rhn/kickstarts/snippets`**. Cobbler accesses snippets in this structure via a symbolic link in **`/var/lib/cobbler/snippets`** callled **`spacewalk`** — thus **`/var/lib/cobbler/snippets/spacewalk`**. Satellite's RPMs expect Cobbler's kickstart and snippet directories to be in their default locations — it is not recommended to alter them.

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

A. Revision History

Revision 1.0