Red Hat Network Satellite 5.3

Deployment Guide

Red Hat Network Satellite

Red Hat Network Satellite 5.3 Deployment Guide
Red Hat Network Satellite
Edition 1

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

Welcome to the RHN Satellite Deployment Guide.

Introduction

Welcome to the RHN Satellite Deployment Guide.

1. More to Come

The *RHN Satellite Deployment Guide* is constantly expanding as new features are launched. HTML and PDF versions of this and other manuals are available within the **Help** section of the RHN Satellite website and at *http://www.redhat.com/docs/manuals/satellite*.

Note

Although this manual reflects the most current information possible, read the *RHN Satellite Release Notes* for information that may not have been available prior to the finalization of the documentation. The notes can be found on the **Help** page of the RHN Satellite Web interface and at *http://www.redhat.com/docs/manuals/satellite/*.

The following RHN documentation has been translated for this RHN Satellite release: RHN Satellite Reference Guide, RHN Satellite Installation Guide, RHN Client Configuration Guide, RHN Channel Management Guide, and RHN Satellite Release Notes. Translated documentation is available at *http://www.redhat.com/docs/* under **Red Hat Network Satellite**.

1.1. Send in Your Feedback

If you would like to make suggestions about the *Red Hat Network Satellite Reference Guide*, please submit a report in Bugzilla (*http://bugzilla.redhat.com/bugzilla/*) against the component Documentation_Deployment_Guide (Product: Red Hat Network Satellite, Version: 531).

Part I. Satellite Administration

This part describes some general practices for Satellite administration, including backing up and restoring Satellite databases, user and organization administration, and more.

Users

The **Users** tab at the top of the Satellite navigation bar allows administrators to manage Satellite users. These pages enable you to grant and edit permissions for those who administer your system groups. Click on the **User List** to modify users within your organization.

1.1. Adding Users

To add new users to your organization, perform the following steps:

1. Click the **create new user** link on the right corner of the **Users** page. You are then presented with the **Create User** page.

2. Create a **Desired Login** name for the user. The login name must be *at least* five characters long.

3. Create and re-enter a **Desired Password** for the user.

4. Enter the name of the user in the **First, Last Name** text box, assigning a prefix (such as Mr., Mrs., or Dr.) from the available drop-down menu.

5. Enter an **Email** address for the user.

Once all fields are complete, select the **Create Login** button. Upon creation of the user, the Satellite sends an email to the specified address and then redirects you to the **Users** -> **User List** page. If you want to select permissions and options for the newly created user, select their name from the list. Doing so displays the **User Details** page for that user, which provides several subtabs of options from which to choose.

1.2. Deleting and Deactivating Users

The **User Details** page allows administrators to manage the permissions and activity of all users. Included in the **User Details** page is the ability to delete or deactivate users.

Satellite administrators may deactivate or delete users from their systems, or users may deactivate their own accounts.

1.2.1. Deactivating Users

Deactivated users cannot log in to the Satellite interface, nor may they schedule any actions. Administrators may not be deactivated until that role is removed from their account. Actions scheduled by a user prior to their deactivation remain in the action queue. For added flexibility, deactivated users may be reactivated by administrators.

To deactivate a user:

1. Navigate to the user's **User Details** tab.

2. Verify that the user is not a Satellite administrator. If they are, uncheck the box to the left of that role and click the **Submit** button in the lower right of the screen.

3. Click the **deactivate user** link.

4. Click the **Deactivate User** button to confirm.

The user no longer appears in the **Active Users** list. Click the **Deactivated** link from the **User List** menu to view the deactivated user.

At any time, you can reactivate the user from the **Deactivated** list by clicking the checkbox next to the user and clicking **Reactivate**.

1.2.2. Deleting Users

Administrators can delete users, but any adminstrator roles that the was assigned must be removed before that user may be deleted.

Warning

User deletion is irreversible; perform this action carefully. Consider disabling the user first in order to assess the effect deletion will have on your Satellite infrastructure.

To delete a user:

1. Navigate to the user's **User Details** tab.
2. Verify that the user is not a Satellite administrator and remove that role if necessary.
3. Click the **delete user** link in the upper right.
4. Click the **Delete User** button to permanently delete the user.

1.3. User Management

The **User Details** tab displays the username, first name, last name, email address, and administrative roles (if any) for the user. All of this information can be modified simply by making any changes and clicking the **Update** button. When changing a user's password, you will see only asterisks as you type for security reasons.

1.3.1. Assigning User Roles

To delegate responsibilities within your organization, Satellite provides several roles with varying degrees of responsibility and access. This list describes the permissions of each and the differences between them:

- **RHN Satellite Administrator** — This is a special role that deals with Satellite administrative tasks, such as creating organizations, managing subscriptions, configuring Proxy settings, and more.
- **Organization Administrator** — This role can perform management functions for a Satellite organization. Organization administrators can manage users, systems, channels, and more in the context of their particular organization.
- **Activation Key Administrator** — This role is designed to manage your organization's collection of activation keys. This person can create, modify, and delete any key within your overarching account.
- **Channel Administrator** — This role has complete access to the software channels and related associations within your organization. It requires RHN Satellite or RHN Proxy Server. This person may change the base channels of systems, make channels globally subscribable, and create entirely new channels.

- **Configuration Administrator** — This role enables the user to manage the configuration of systems in the organization using the RHN Satellite interface.
- **Monitoring Administrator** — This role allows for the scheduling of probes and oversight of other monitoring infrastructure. This role is available only on Monitoring-enabled Satellites.
- **System Group Administrator** — This role is one step below Organization Administrator in that it has complete authority over the systems and system groups to which it is granted access. This user can create new system groups, delete any assigned systems groups, add systems to groups, and manage user access to groups.

While it is possible for one Satellite administrator to remove Satellite administrator rights from another user, it is impossible to remove Satellite administrator rights from the sole remaining Satellite administrator. It is possible to remove your own Satellite administrator privileges so long as you are not the last Satellite administrator.

To assign a user a new role, select the appropriate checkbox. Remember that Organization administrators are automatically granted administration access to all other roles, signified by grayed-out checkboxes. When satisfied with the changes, click **Update**.

Satellite Operation Guidance

Red Hat Network (RHN) is the environment for system-level support and management of Red Hat systems and networks of systems. Red Hat Network brings together the tools, services, and information repositories needed to maximize the reliability, security, and performance of their systems. To use RHN, system administrators register software and hardware profiles, known as System Profiles, of their client systems with Red Hat Network. When a client system requests package updates, only the applicable packages for the client are returned.

RHN Satellite Server allows organizations to utilize the benefits of Red Hat Network without having to provide public Internet access to their servers or other client systems. System Profiles are stored locally on the customer's RHN Satellite Server. The Red Hat Network website is served from a local web server and is only accessible to systems that can reach the Satellite. All package management tasks, including Errata Updates, are performed through your network.

RHN Satellite Server provides a solution to organizations requiring absolute control over and privacy of the maintenance and package deployment of their servers. It allows Red Hat Network customers the greatest flexibility and power in keeping systems secure and updated. Modules can be added to the Satellite Server to allow extra functionality. This document provides guidance on several common supporting operations which are essential when running RHN Satellite Server.

2.1. Automating Synchronization

Manually synchronizing the RHN Satellite Server repository with Red Hat Network can be an arduous task. In addition, staff levels tend to be highest at peak usage times. For this reason, you should automate synchronization in late evening or early morning to better balance load and ensure quick synchronization. Synchronization should occur randomly for best performance.

This automation can be set easily by the addition of a simple cron job. To do this, edit the crontab as root:

```
crontab -e
```

This opens the crontab in a text editor, by default vi. Another editor can be used by first changing the EDITOR variable, e.g., export EDITOR=gedit. Once opened, use the first five fields (minute, hour, day, month, and weekday) to schedule the synchronization.

Remember, hours use the 24-hour clock. Edit the **`crontab`** to include random synchronization:

```
0 1 * * * perl -le 'sleep rand 9000' && satellite-sync --email >/dev/null 2>1
```

This particular job will run randomly between 1:00 AM and 3:30 AM system time each night and discards stdout and stderr from cron to prevent duplicating the more easily read message from satellite-sync. Options other than --email can also be included. Once you exit from the editor, the modified crontab is installed immediately.

2.2. Backup and Restore Routines

To backup, verify, and restore a Satellite system, perform the tasks in this chapter.

2.2.1. Backing up the Embedded Database

To backup the Embedded RHN Satellite database, perform the following tasks:

1. Stop the Satellite Server:

   ```
   rhn-satellite stop
   ```

2. Create the Backup using **`db-control`**. Switch to the Oracle user before executing the db-control utility. Start the backup with the following command:

   ```
   su - oracle
   db-control backup [directory]
   ```

   Replace *`directory`* with the absolute path to the location where you want to store your database backup. The process will take several minutes.

3. Resume the Satellite. Switch back to the root user and start up the Satellite:

   ```
   exit
   rhn-satellite start
   ```

4. Verify the Backup. The **`examine`** option of db-control, issued as the Oracle user, conducts a quick check of the backup time stamp and determines any missing files:

   ```
   su - oracle
   db-control examine [directory]
   ```

   The **`verify`** option of db-control, also issued as the Oracle user, conducts a thorough review. This includes checking the md5sum of each of the files in the backup:

   ```
   db-control verify [directory]
   ```

   If the verification returns as successful, it is safe to rely on the contents of *`directory`* to restore the database.

Note

As with embedded databases, it is recommended for external database users of Satellite to perform periodic backups as well. Consult your external database administrator for more information on supported backup procedures.

2.2.2. Backing up System Files

In addition to the database, also a number of system files and directories should be backed up. Below is the minimum list files and directories that should be backed up:

- /rhnsat/ (*Embedded Database; never to be backed up while the database is running*)

- /etc/sysconfig/rhn/
- /etc/rhn/
- /etc/sudoers
- /etc/tnsnames.ora
- /var/www/html/pub/
- /var/satellite/redhat/[0-9]*/ (*custom RPMs*)
- /root/.gnupg/
- /root/ssl-build/
- /etc/dhcpd.conf
- /etc/httpd/
- /tftpboot/
- /var/lib/cobbler/
- /var/lib/nocpulse/
- /var/lib/rhn/kickstarts/
- /var/www/cobbler/

If possible, back up **`/var/satellite/`**, as well. In case of failure, this will save lengthy download time. Since **`/var/satellite/`** (specifically **`/var/satellite/redhat/NULL/`** and **`/var/satellite/rhn/`**) is primarily a duplicate of Red Hat's RPM repository, it can be regenerated with satellite-sync. Red Hat recommends the kickstart tree be backed up. In the case of disconnected satellites, **`/var/satellite/`** must be backed up.

Backing up only these files and directories would require reinstalling the RHN Satellite Server ISO RPMs and re-registering the Satellite. In addition, Red Hat packages would need to be resynchronized using the satellite-sync tool. Finally, you would have to reinstall the **`/root/ssl-build/rhn-org-httpd-ssl-key-pair-MACHINE_NAME-VER-REL.noarch.rpm`**. Another method would be to back up all of the files and directories mentioned above but reinstall the RHN Satellite without re-registering it. During the installation, cancel or skip the RHN registration and SSL certificate generation sections.

The final and most comprehensive method would be to back up the entire machine. This would save time in downloading and reinstalling but would require additional disk space and back up time.

Regardless of the backup method used, when you restore the Satellite from a backup, you must run the following command to schedule the recreation of search indexes the next time the **`rhn-search`** service is started:

```
/etc/init.d/rhn-search cleanindex
```

2.2.3. Restoring the Embedded Database

RHN Database Control makes Embedded Database restoration a relatively simple process.

1. First, stop the database and related services with the following command:

   ```
   rhn-satellite stop
   ```

2. Restore the Backup with db-control — Switch to the Oracle user and use the following command, replacing *`directory`* with the directory that contains the backup:

   ```
   su - oracle
   db-control restore [directory]
   ```

   This not only restores the Embedded Database, but first verifies the contents of the backup directory using md5sums.

3. Resume the Satellite — Once the restoration is complete, return to root user mode and restart the database and related services:

   ```
   exit
   rhn-satellite start
   ```

2.2.4. Backup Automation

Red Hat strongly recommends scheduling the backup process automatically using cron jobs. In the following example crontab excerpt and scripts, the database is backed up at 3 AM and the resulting files are moved to a separate repository at 6 AM. The following is an example excerpt from root **`crontab`** file:

```
0 3 * * * backup-db.sh
0 6 * * * move-files.sh
```

The following is an example of the aforementioned **`backup-db.sh`** script:

```
#!/bin/bash
{
/usr/sbin/rhn-satellite stop
su - oracle -c'
d=db-backup-$(date "+%F");
mkdir -p /tmp/$d;
db-control backup /tmp/$d
';
/usr/sbin/rhn-satellite start
} &> /dev/null
```

And, here is an example of **`move-files.sh`** (replace *`[destination]`* with the path to the backup directory:

```
#!/bin/bash
rsync -avz /tmp/db-backup-$(date "+%F") [destination] &> /dev/null
```

Or:

```
#!/bin/bash
scp -r /tmp/db-backup-$(date "+%F") [destination] &> /dev/null
```

In addition, a clean up script to remove older backup directories should be utilized to prevent the storage from filling up.

2.3. Monitoring the Satellite

In Oracle databases, check on a regular basis that the tablespaces have free space:

```
su - oracle
db-control report
```

This will print out current tablespace usage, for example:

```
Tablespace    Size  Used Avail  Use%
DATA_TBS      4.8G 3.9G  996M   80%
SYSTEM        250M 116M  133M   46%
TOOLS         128M 3M    124M   2%
UNDO_TBS      1000M 61M  938M   6%
USERS         128M 64K   127M   0%
```

If a tablespace is filling up – extend it by running:

```
db-control extend tablespace
```

To verify that all Satellite processes are working run:

```
rhn-satellite status
```

2.4. Implementing PAM Authentication

As security measures become increasingly complex, administrators must be given tools that simplify their management. RHN Satellite supports network-based authentication systems via Pluggable Authentication Modules (PAM). PAM is a suite of libraries that helps system administrators integrate the Satellite with a centralized authentication mechanism, thus eliminating the need for remembering multiple passwords.

RHN Satellite supports LDAP, Kerberos, Directory Server and other network-based authentication systems. To enable the Satellite to use PAM and your organization's authentication infrastructure, follow the steps below.

Note

To ensure that PAM authentication functions properly, install the **`pam-devel`** package.

Set up a PAM service file (usually **`/etc/pam.d/rhn-satellite`**) and have the Satellite use it by adding the following line to **`/etc/rhn/rhn.conf`**:

```
pam_auth_service = rhn-satellite
```

This assumes the PAM service file is named **`rhn-satellite`**.

To enable a user to authenticate against PAM, select the checkbox labeled **Pluggable Authentication Modules (PAM)**. It is positioned below the password and password confirmation fields on the **Create User** page.

As an example, for a Red Hat Enterprise Linux 5 i386 system, to authenticate against Kerberos you can add the following to **`/etc/pam.d/rhn-satellite`**:

```
#%PAM-1.0
auth        required      pam_env.so
auth        sufficient    pam_krb5.so no_user_check
auth        required      pam_deny.so
account     required      pam_krb5.so no_user_check
```

Note that changing the password on the RHN website changes only the local password on the Satellite server, which may not be used at all if PAM is enabled for that user. In the above example, for instance, the Kerberos password will not be changed.

For LDAP authentication on 32-bit systems, add the following lines to the **`/etc/pam.d/rhn-satellite`** file:

```
#%PAM-1.0
auth         required      /lib/security/pam_env.so
auth         sufficient    /lib/security/pam_ldap.so no_user_check
auth         required      /lib/security/pam_deny.so
account      required      /lib/security/pam_ldap.so no_user_check
```

For LDAP support on 64-bit Satellite servers, add the following lines:

```
#%PAM-1.0
auth      required      /lib64/security/pam_env.so
auth      sufficient    /lib64/security/pam_ldap.so no_user_check
auth      required      /lib64/security/pam_deny.so
account   required      /lib64/security/pam_ldap.so no_user_check
```

For more information about configuring PAM, refer to the Chapter entitled "Pluggable Authentication Modules (PAM)" in the *Red Hat Enterprise Linux Deployment Guide*.

2.5. RPM Building

As part of automated installations administrators will often deploy custom applications not provided by Red Hat, such as backup and monitoring software. In order to do this, this software must be packaged as RPMs. An RPM build environment can be set up on a system running Red Hat Enterprise Linux.
It should be noted that the build system must contain the same version of packages which are used in target systems. This means that a RHEL 4 system must be used to build RPMs for RHEL 4 based systems and a RHEL 5 system for RHEL 5 RPMs. The package rpm-build must be installed on the build system as a minimum requirement but also additional packages like compilers and libraries might be needed. Production ready RPM packages should be signed with a GPG key allowing to verify the origin and integrity of packages. The passphrase of the GPG key used for signing RPMs should be known only to a trusted group of administrators.

2.5.1. `rpmbuild`

A user account for building, e.g., **`rpmbuild`** should be created to allow several administrators to share the build environment and the GPG key. It is recommended that this non-privileged user is used when building RPMs. The home directory for the user, /home/rpmbuild by default, should contain the file .rpmmacros with at least the following content (where _gpg_name must match the name for the GPG key used for signing RPMs):

```
%_topdir            %(echo $HOME)/rpmbuild
%_signature         %gpg
%_gpg_name          rpmbuild <rpmbuild@example.com>
```

The directory listing for the defined top level directory (/home/rpmbuild/rpmbuild in the example above) must have the same directory layout that is present under /usr/src/redhat. Appendix B contains a very basic RPM spec file which can be used a basis when starting to craft real spec files needed locally.

2.5.2. GPG Key for Signing RPMs

The following commands will initiate GPG key creation and export it in a format suitable for distributing to client systems. The created key should be stored safely and backed up, and its passphrase should be known only by trusted administrators.

```
mkdir -p ~/.gnupg
gpg --gen-key
gpg --list-keys --fingerprint
gpg --export --armor "rpmbuild <rpmbuild@example.com>" > EXAMPLE-RPM-GPG-KEY
```

To import this key to the RPM database to allow RPM origin and integrity verification, the following command must be run as root on all target systems (naturally this should happen automatically during client installations):

```
rpm --import EXAMPLE-RPM-GPG-KEY
```

Once an RPM has been created it must be signed with the GPG key and uploaded to a correct channel:

```
rpm --resign package.rpm
rhnpush --server=http[s]://satellite.server/APP package.rpm --channel=custom-channel-name
```

The following commands will verify an RPM package located in the current directory:

```

rpm –qip pakcage.rpm
rpm -K package.rpm
```

2.5.3. RPM Spec File Example

The following is a basic example of an RPM spec file. When building, it should be located in the directory SPECS under the _topdir as defined in user's .rpmmacros file and the corresponding source and patch files should in the SOURCES directory.

```

Name: foo
Summary: The foo package does foo
Version: 1.0
Release: 1
License: GPL
Group: Applications/Internet
URL: http://www.example.org/
Source0 : foo-1.0.tar.gz
Buildroot: %{_tmppath}/%{name}-%{version}-%{release}-root
Requires: pam
BuildPrereq: coreutils
%description
This package performs the foo operation.
%prep
%setup -q
%build
%install
mkdir -p %{buildroot}/%{_datadir}/%{name}
cp -p foo.spec %{buildroot}/%{_datadir}/%{name}
%clean
rm -fr %{buildroot}
%pre
# Add user/group here if needed
%post
/sbin/chkconfig --add food
%preun
if [ $1 = 0 ]; then # package is being erased, not upgraded
    /sbin/service food stop > /dev/null 2>&1
    /sbin/chkconfig --del food
fi
%postun
if [ $1 = 0 ]; then # package is being erased
    # Any needed actions here on uninstalls
else
    # Upgrade
    /sbin/service food condrestart > /dev/null 2>&1
fi
%files
%defattr(-,root,root)
%{_datadir}/%{name}
%changelog
* Mon Jun 16 2003 Some One <one@example.com>
- fixed the broken frobber (#86434)
```

2.6. Kickstart

System kickstarting is essential part of automated installation and efficient system provisioning. PXE combined with remote hardware management provides fully automated solution for kickstarting. However, parts of the network might not have PXE/DHCP functionality available so also CD/USB booting might be needed in some cases. The following sections provide step by step instructions on how to create customized boot images for kickstarting clients with CD/PXE/USB using standard Linux utilities from the syslinux package. Some of the steps (beginning with #) expects that they are run either with sudo or as root. Appendix C provides an example of boot menus for different kickstart targets that can be interactively selected when provisioning a system.

2.6.1. Preparing Custom Boot Media

First, locate the standard RHEL CD boot image boot.iso, create work directories for different boot images, and populate them with RHEL boot files:

```
mkdir -p temp cd/isolinux pxe/pxelinux.cfg usb/extlinux usb/temp
mount -o loop boot.iso temp
cp -a temp/isolinux/* cd/isolinux/
cp -a temp/isolinux/* pxe/
cp -a temp/isolinux/* usb/extlinux/
umount temp
chmod -R u+rw cd pxe usb
```

Follow the instructions below to create needed boot images.

2.6.2. Creating CD Boot Media

Creating bootable CD image can done with the following commands after the earlier preparations (note that the backslash "\" represents a continuation of one line at the shell prompt):

```
cd ./cd
cp -p /usr/lib/syslinux/menu.c32 isolinux
vi isolinux/isolinux.cfg
mkisofs -o ./custom-boot.iso -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot \
  -boot-load-size 4 -boot-info-table -J -l -r -T -v -V "Custom RHEL Boot" .
```

Customize any boot parameters and targets in isolinux.cfg as needed for CD booting.

2.6.3. Preparing PXE Boot Directory

Creating a directory structure suitable for PXE booting can be done with the following commands after the previous preparations:

```
cd ../pxe
cp -p /usr/lib/syslinux/menu.c32 .
mv isolinux.cfg pxelinux.cfg/default
rm -f isolinux.bin TRANS.TBL
vi pxelinux.cfg/default
cp -p /usr/lib/syslinux/pxelinux.0 .
```

Customize any boot parameters and targets in pxelinux.cfg/default as needed for PXE booting.

2.6.4. Creating USB Boot Media

Creating bootable USB image can done with the following commands after the preparations above. Be extremely careful when carrying out these command as root (required for most critical parts) since they access device files and may irrecoverably damage your system! The example below uses /dev/loop0 for mounting, make sure you use the device which the command losetup -f prints.

```
cd ../usb
cp -p /usr/lib/syslinux/menu.c32 extlinux/
mv extlinux/isolinux.cfg extlinux/extlinux.conf
rm -f extlinux/isolinux.bin extlinux/TRANS.TBL
vi extlinux/extlinux.conf
dd if=/dev/zero of=./custom-boot.img bs=1024 count=30000
losetup -f
losetup /dev/loop0 ./custom-boot.img
fdisk /dev/loop0
```

Under **`fdisk`**, create one primary bootable partition by following the sequence of keypress commands entered within the **`fdisk`** interface:

```
n, p, 1, Enter, Enter, a, 1, p, w
```

Now, finish the process:

```
dd if=/usr/lib/syslinux/mbr.bin of=/dev/loop0
kpartx -av /dev/loop0
mkfs.ext2 -m 0 -L "Custom RHEL Boot" /dev/mapper/loop0p1
mount /dev/mapper/loop0p1 temp
rm -rf temp/lost+found
cp -a extlinux/* temp/
extlinux temp
umount temp
kpartx -dv /dev/loop0
losetup -d /dev/loop0
sync
```

Customize the boot targets and boot parameters in extlinux.conf as needed for USB booting. To transfer the image to a USB stick, insert the stick into your computer, check from dmesg output its device name (/dev/sdb in the example below), unmount it, and transfer the image:

```
umount /dev/sdb
dd if=./custom-boot.img of=/dev/sdb
```

2.6.5. Kickstart Boot Menu Example

The following files, when placed in the same directory as the menu helper program menu.c32 will present a boot menu which can be interactively navigated and edited byusing the arrow keys and the tabulator.

```
Main Menu
  DEFAULT menu.c32
```

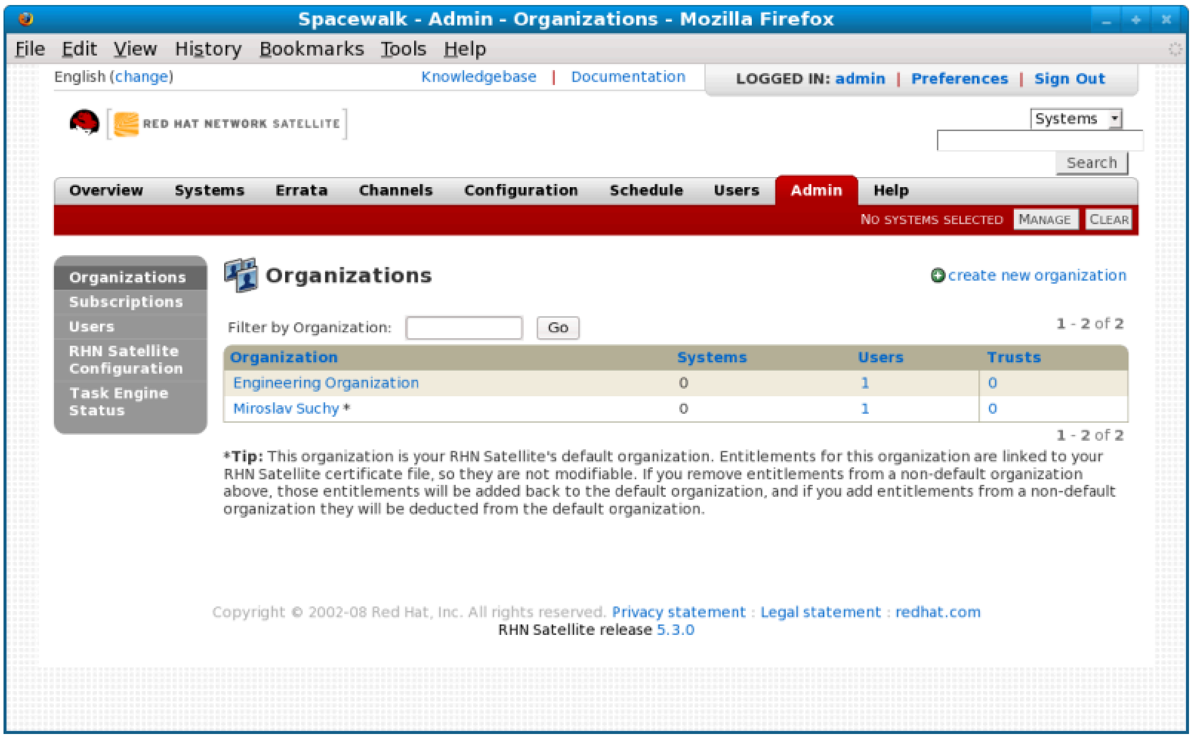
```
MENU TITLE Red Hat Enterprise Linux Installation
TIMEOUT 100
ALLOWOPTIONS 0
NOESCAPE 1
PROMPT 0
MENU MASTER PASSWD password
LABEL RHEL
 MENU LABEL TS 1.0 RHEL 5.4 x86_64
 KERNEL menu.c32
 APPEND rhel5.cfg
# ISO/EXTLINUX
LABEL LOCAL
 MENU LABEL Boot from local hard disk
 LOCALBOOT 0
# PXELINUX
# LABEL LOCAL
# MENU LABEL Boot from local hard disk
# KERNEL chain.c32
# APPEND hd0 0
```

Multiple Organizations

RHN Satellite supports the creation and management of *multiple organizations* within one Satellite installation, allowing for the division of systems, content, and subscriptions across different organizations or specific groups. This chapter summarizes the basic setup tasks and concepts of multiple organization creation and management within RHN Satellite.

3.1. Admin -> Organizations

The **Organizations** Web interface allows administrators to view, create, and manage multiple Satellite organizations. Satellite administrators can allocate software and system entitlements across various organizations, as well as control an organization's access to systems management tasks.


Spacewalk - Admin - Organizations - Mozilla Firefox
File Edit View History Bookmarks Tools Help
English (change)
Knowledgebase | Documentation
LOGGED IN: admin | Preferences | Sign Out
RED HAT NETWORK SATELLITE
Systems
Search
Overview Systems Errata Channels Configuration Schedule Users Admin Help
No systems selected Manage Clear
Organizations
Subscriptions
Users
RHN Satellite Configuration
Task Engine Status
create new organization
Filter by Organization: Go

Organizations

1 - 2 of 2

Organization	Systems	Users	Trusts
Engineering Organization	0	1	0
Miroslav Suchy *	0	1	0

1 - 2 of 2

***Tip:** This organization is your RHN Satellite's default organization. Entitlements for this organization are linked to your RHN Satellite certificate file, so they are not modifiable. If you remove entitlements from a non-default organization above, those entitlements will be added back to the default organization, and if you add entitlements from a non-default organization they will be deducted from the default organization.

Copyright © 2002-08 Red Hat, Inc. All rights reserved. Privacy statement : Legal statement : redhat.com
RHN Satellite release 5.3.0

Figure 3.1. Admin

The **Organizations** page contains a listing of organizations across the Satellite, with both User and System counts assigned to each organization. The **Organizations** page also features a **Trusts** page for any organizational trusts established. Refer to *Chapter 4, Organizational Trusts* for more information about establishing organizational trusts.

3.1.1. Admin -> Organizations -> Details

Clicking on an organization displays the **Details** page, where administrators are provided a summary of various aspects of the organization.

- **Active Users** — The number of users in the organization
- **Systems** — The number of systems subscribed to the organization.
- **System Groups** — The number of groups subscribed to the organization.

- **Activation Keys** — The number of activation keys available to the organization.
- **Kickstart Profiles** — The number of kickstart profiles available to the organization.
- **Configuration Channels** — The number of Configuration Channels available to the organization.

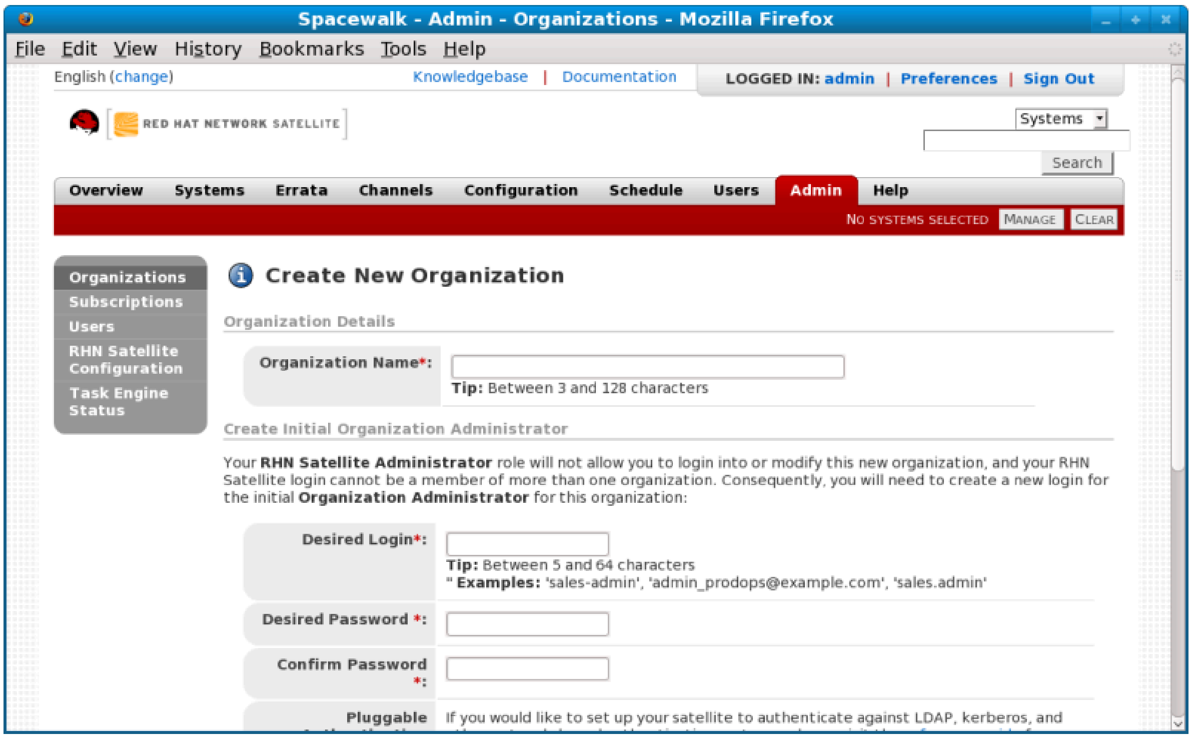
From this page, you can delete the organization by clicking the **Delete Organization** link.

The **Details** page also contains three subtabs: **Users**, **Subscriptions**, and **Trusts**.

3.2. Creating an Organization

The **Create New Organization** page in the RHN Satellite web interface can be accessed by proceeding to **Admin** -> **Organizations** -> **Create New Organization**.

Satellite administrators can create new organizations, assign administrators for those organizations, and assign entitlements. Organization administrators can assign groups, systems, and users for their organization. This division allows organizations can perform administrative tasks on their own without affecting other organizations.


Spacewalk - Admin - Organizations - Mozilla Firefox
File Edit View History Bookmarks Tools Help
English (change)
Knowledgebase | Documentation
LOGGED IN: admin | Preferences | Sign Out
RED HAT NETWORK SATELLITE
Systems
Search
Overview Systems Errata Channels Configuration Schedule Users Admin Help
No systems selected Manage Clear
Organizations
Subscriptions
Users
RHN Satellite Configuration
Task Engine Status

Create New Organization

Organization Details

Organization Name*:
Tip: Between 3 and 128 characters

Create Initial Organization Administrator

Your **RHN Satellite Administrator** role will not allow you to login into or modify this new organization, and your RHN Satellite login cannot be a member of more than one organization. Consequently, you will need to create a new login for the initial **Organization Administrator** for this organization:

Desired Login*:
Tip: Between 5 and 64 characters
" **Examples:** 'sales-admin', 'admin_prodops@example.com', 'sales.admin'

Desired Password *:

Confirm Password *:

Pluggable
If you would like to set up your satellite to authenticate against LDAP, kerberos, and

Figure 3.2. Create New Organization

1. Input the **Organization Name** in the provided text box. The name should be between 3 and 128 characters.
2. Create an administrator for the organization:
    a. Enter a **Desired Login** for the organization administrator, which should be between 3 and 128 characters long.
    b. Create a **Desired Password** and **Confirm** the password.
    c. Type in the **Email** address for the organization administrator.

   d. Enter the **First Name** and **Last Name** of the organization administrator.

3. Click the **Create Organization** button to complete the process.

Once the new organization is created, the **Organizations** page will display with the new organization listed.

Tip

Satellite Administrators should consider reserving the organization 1 Organization Administrator account for themselves to have the option of logging into this organization for various reasons. If your Satellite is configured for PAM authentication, avoid using PAM accounts for the one Satellite administrative organization administrator account in new organizations. Instead, create a Satellite-local account for organization administrators and reserve PAM-authenticated accounts for Satellite logins with less elevated privileges in order to discourage users to frequently log into the Satellite with elevated privileges, as the potential for making mistakes is higher using these accounts.

Additionally, consider creating a login name for the Organization Administrator account that describes (for example, **`orgadmin-mktg`** or **`eng-dept-admin`**), to match admin login names with the organization.

3.3. Managing Entitlements

One important task after creating a new organization is to assign management entitlements to the new organization. Management system entitlements are a base requirement for an organization to function on the Satellite. The number of management entitlements allocated to an organization is equivalent to the maximum number of systems that may register to that organization on the Satellite, regardless of the number of software entitlements available. For example, if there are 100 Red Hat Enterprise Linux Client entitlements but only 50 management system entitlements to an organization, only 50 systems are able to register to that organization.

You should also grant RHN Tools software channel entitlements to each organization. The RHN Tools channel contains various client software required for extended Satellite functionality, such as clients necessary for configuration management and kickstart support as well as the **`rhn-virtualization`** package, which is necessary for the entitlements of Xen virtual guests to be counted correctly corresponding to the number of Red Hat Enterprise Linux subscriptions to which they are associated.

Access the **Subscriptions** tab by clicking **Admin** -> **Organizations** -> **Details** -> **Subscriptions**.

The **Subscriptions** tab has two subtabs for managing the software channel and system entitlements for the organization.

3.3.1. Admin -> Subscriptions -> Software Channel Entitlements

The **Software Channel Entitlements Across Satellite** page lists of all entitlements on the Satellite, throughout all organizations, as well their usage. Click on a **Entitlement Name** for a more detailed view.

The **Details** subtab for the software channel entitlement contains information about the software channel access granted when subscribed to the entitlement.

The **Organizations** subtab allows Satellite administrators to adjust the number of software channels available to each organization. Type in the number (within the range listed in **Possible Values**) and click the **Update** button for that organization.

Note

Organization Administrators that create a custom channel can only use that channel within their organization unless an Organizational Trust is established between the organizations that want to share the channel. For more information about organizational trusts, refer to *Chapter 4, Organizational Trusts*.

The **Organizations** subtab also contains broad usage information in the **System-Wide Entitlement Usage** section, including:

- **Total** — The total number of channel entitlements for the Satellite.
- **Available** — The number of entitlements currently available for allocation.
- **Usage** — The number of entitlements currently in use by all organizations, compared to the total number of entitlements allocated.

For example, if the **Total** column is 100 and the **Available** column is 70, that means 30 entitlements are allocated for organizations. The **Usage** column shows how many of those 30 allocated entitlements are in use by organizations besides the base organization. So if the **Usage** column reads *`24 of 30 (80%)`*, that means 24 channel entitlements are distributed to Satellite organizations (other than organization 1) out of 30 total allocated.

3.3.2. Admin -> Subscriptions -> System Entitlements

The **System Entitlements Across Satellite** page lists all system entitlements on this Satellite, across all organizations, as well as their usage. Click on the entitlement's name for more details about it.

System entitlements include **Management**, **Provisioning**, **Monitoring**, **Virtualization**, and **Virtualization Platform**. Enter the number of allocations of each system entitlement in the text box, not to exceed the limit indicated in the **Possible Values**.

The **Details** subtab for the system entitlement contains information about the entitlement and what access it grants.

The **Organizations** subtab allows Satellite administrators to adjust the number of system entitlement allocations available to each organization. Type in the number (within the range listed in **Possible Values**) and click the **Confirm Changes** button for that organization.

The **Organizations** subtab for the system entitlement also contains broad usage information in the "Satellite-Wide Entitlement Usage" section, including:

- **Total Allocated** — The number of total entitlements available for the entire Satellite.
- **Entitlement Usage** — The number of entitlements currently being used.
- **Organization Usage** shows the number of organizations that have access to the entitlement.

3.4. Configuring Systems in an Organization

Now that an organization has been created and requisite entitlements assigned to it, you can then assign systems to each organization.

There are two basic ways to register a system against a particular organization:

1. Registering Using username and Password — If you provide a username and password created for a specified organization, the system will be registered to that organization. For example, if **`user-123`** is a member of the **Central IT** organization on the Satellite, the following command on any system would register that system to the **Central IT** organization on your Satellite:

   ```
   rhnreg_ks --username=user-123 --password=foobaz
   ```

   Note

   The `--orgid` (for Red Hat Enterprise Linux 4 and 5) and `--orgpassword` (in RHEL 4) parameters in **`rhnreg_ks`** are *not related* to Satellite registration or RHN Satellite's multiple organizations support.

2. Registering Using An Activation Key — You can also register a system to an organization using an activation key from the organization. Activation keys will register systems to the organization in which the activation key was created. Activation keys are a good registration method to use if you want to allow users to register systems into an organization without providing them login access to that organization. If you want to move systems between organizations, you may also automate the move with scripts using the activation keys.

   Note

   Activation keys have a new format since RHN Satellite 5.1.0, so the first few characters of the activation key are used to indicate which organization (by ID number) owns the key.

3.5. Admin -> Users

The **Users Across Satellite** page contains a list of all users on the Satellite, throughout all organizations.

Note

You are only able to modify the details of organization users if you are logged in as that Organization Administrator.

Clicking the **Username** displays the **User Details** page.

3.5.1. Admin -> Organizations -> Details -> Users

The **Users** subtab lists the users assigned to the organization, including their real names, email address, and a check mark indicating that the user is an administrator of the organization.

If you are the Organization Administrator, you can click the username to display the **User Details** page for the user.

Note

You must be logged in as the Organization Administrator to edit the User details for an organization. The Satellite Administrator cannot edit user details for organization users.

Organizational Trusts

IT deployments are not one-dimensional. Within any organization, the IT infrastructure truly needs to be managed in a multitenant / multi-organizational structure. Whether the division is bureaucratic (organizational departments, offices, subsidiaries) or functional (Web apps, databases, desktops, business processes) Red Hat Network Satellite enables administrators to divide their deployments into organized containers. These containers (or *organizations*) assist in maintaining clear separation of purpose and ownership of systems and the content deployed to those systems.

Red Hat Network Satellite 5.1 introduced the creation and management of multiple organizations within one Satellite installation, allowing for the division of systems, content, and subscriptions across these different organizations, departments, and other specified groups. Red Hat Network Satellite 5.3 expanded multiple organization support to include organizational trusts that allow sharing custom channel content and migration of systems between organizations. This guide will help you understand and make the most of these new capabilities.

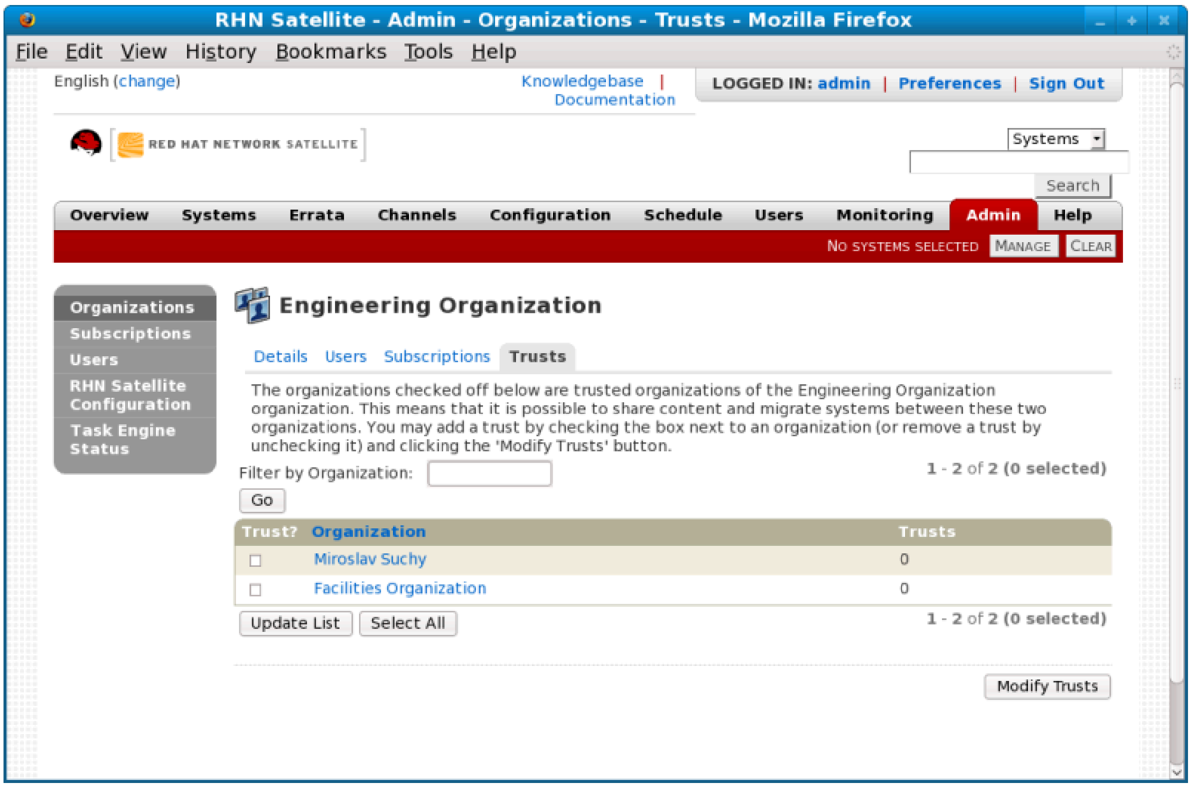
4.1. Establishing an Organizational Trust

A Satellite Administrator can create a trust between two or more organizations. To do this, perform the following steps:

1. Click the **Organizations** link on the side menu on the **Admin** main page.

2. Click the name of one of the organizations and within the **Details** page, click the **Trusts** subtab.

3. On the **Trusts** subtab, there is a listing of all the other trusts on the RHN Satellite. Here you may use the **Filter by Organization** text box to narrow down a long list of organizations to a specific subset.


RHN Satellite - Admin - Organizations - Trusts - Mozilla Firefox
File Edit View History Bookmarks Tools Help
English (change)
Knowledgebase | Documentation
LOGGED IN: admin | Preferences | Sign Out
RED HAT NETWORK SATELLITE
Systems
Search
Overview Systems Errata Channels Configuration Schedule Users Monitoring Admin Help
No systems selected MANAGE CLEAR
Organizations
Subscriptions
Users
RHN Satellite Configuration
Task Engine Status
Engineering Organization
Details Users Subscriptions Trusts

The organizations checked off below are trusted organizations of the Engineering Organization organization. This means that it is possible to share content and migrate systems between these two organizations. You may add a trust by checking the box next to an organization (or remove a trust by unchecking it) and clicking the 'Modify Trusts' button.

Filter by Organization: Go — 1 - 2 of 2 (0 selected)

Trust?	Organization	Trusts
☐	Miroslav Suchy	0
☐	Facilities Organization	0

Update List | Select All — 1 - 2 of 2 (0 selected)

Modify Trusts

Figure 4.1. Organizational Trusts

4. Click the checkbox next to the names of the organizations you want to be in the organizational trust with the current organization

5. Click the **Modify Trusts** button

4.2. Organization Trust Relationships

Organizations can share their resources with each other by establishing an organizational trust relationship. Organizational trusts are defined by the Satellite Administrator and implemented by the Organization Administrator. An organizational trust is bi-directional, meaning that once a Satellite Administrator establishes a trust between two or more organizations, the Organization Administrator from each organization is free to share as much or as little of their resources as they need to. It is up to each Organization Administrator to determine what resources to share, and what shared resources from other organizations in the trust relationship to use.

Each individual relationship is unique and mutually exclusive from other org's trust relationship. For example: if the Accounting Org trusts the Finance Org , and the Finance Org trusts the Facilities Org, then the Accounting Org will not trust the Facilities Org unless a separate trust relationship is defined between Accounting and Facilities organizations.

4.3. Sharing Custom Channels

Once an organizational trust has been established, organizations can now share custom software channels with the other organizations in the trust. There are three levels of channel sharing that can be applied to each channel for finer-grained channel access control:

- Private — Make the channel private so that it cannot be accessed by any organizations except the owning organization.
- Protected — Allow the channel to be accessed by specific trusted organizations of your choice.
- Public — Allow all organizations within the trust to access the custom channel.

Trusted Organizations that are granted access to the custom content using the protected or public access modes can allow their client systems to install and update packages from the shared channel. When a Organization Administrator changes the access of a shared channel to a access level that is not allowed for a consuming organization, the systems subscribed to the shared channel will lose subscription access.

Systems that are subscribed to shared content can be unsubscribed automatically when the following actions occur:

- Trust Relationship is removed by the Satellite Administrator
- Organization Administrator changes channel access to private
- Organization Administrator changes channel access to public and does not include the subscribed system's organization in the protected list
- Organization Administrator deletes the shared channel directly
- Organization Administrator deletes the parent channel of a shared child channel

All Red Hat software channels are managed through entitlements. Organization Administrators cannot share Red Hat Channels because they are available to all organizations that have entitlements to those channels. The Satellite administrator is responsible for assigning Red Hat software channel entitlements to each organization.

4.4. System Migration

In addition to sharing software channels, organizations in a trust can migrate systems to other trusted organizations by using a utility, migrate-system-profile, that is executed from the command-line using the systemID and orgID as arguments to specify the system migration and its destination organization. The Satellite Administrator can migrate a system from one trusted organization to any other in the trust. However, Organization Administrators can only migrate a system from their own organization to another in the trust.

The **`migrate-system-profile`** command requires the **`spacewalk-utils`** package to be installed. **`spacewalk-utils`** is installed on Satellite 5.3 by default but may also be installed on other clients that are subscribed to the **rhn-tools** channel. When an organization migrates a system with the **`migrate-system-profile`** command, the system does not carry any of the previous entitlements or channel subscriptions from the source organization. However, the system's history is preserved, and can be accessed by the new Organization Administrator in order to simplify the rest of the migration process, which includes subscribing to a base channel and granting entitlements.

The usage from the command line is the following:

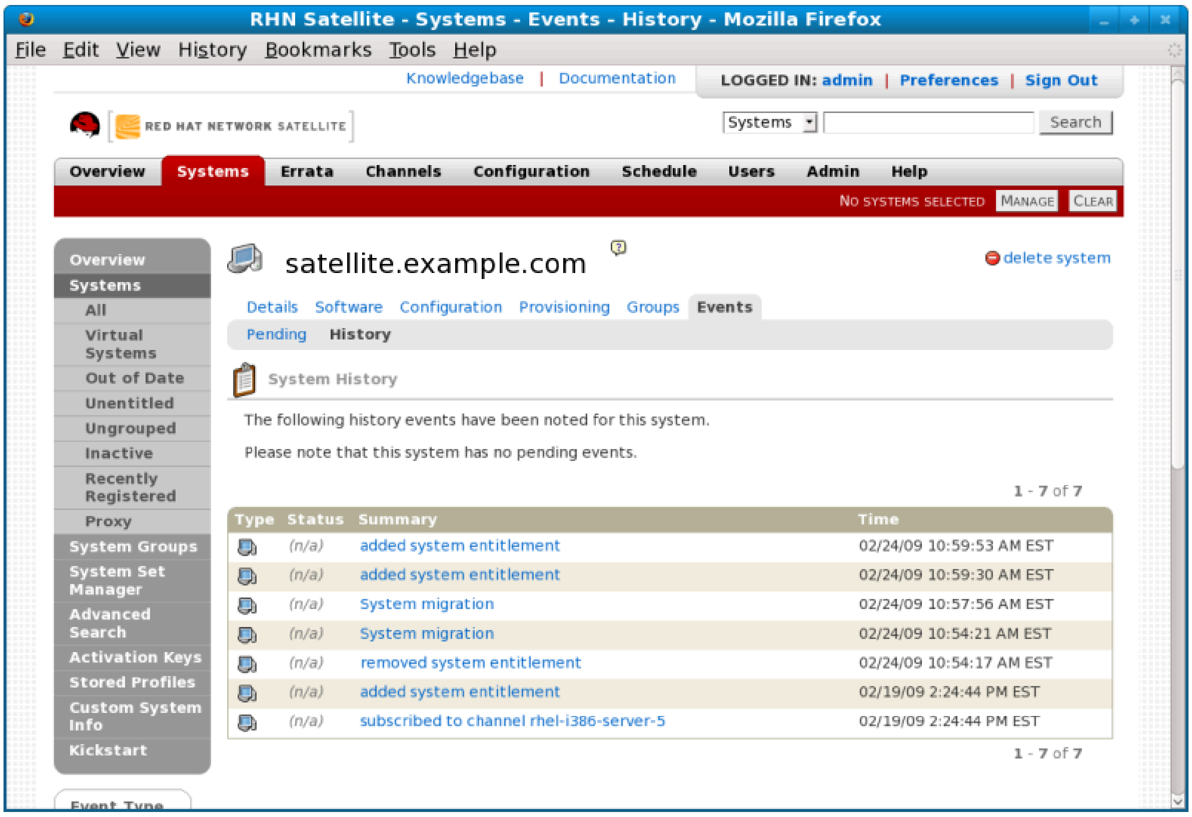
```
migrate-system-profile --satellite {SATELLITE HOSTNAME OR IP} --systemId={SYSTEM ID} --to-org-
id={DESTINATION ORGANIZATION ID}
```

For example, if the Finance department (created as an organization in RHN Satellite with OrgID 2) wants to migrate a workstation (with SystemID 10001020) from the Engineering department, but the Finance Organization Administrator does not have shell access to the RHN Satellite server. The RHN Satellite hostname is **satserver.example.com**. The Finance Organization Administrator would type the following from a shell prompt:

```
migrate-system-profile --satellite satserver.example.com --systemId=10001020 --to-org-id=2
```

The Finance Organization Administrator is then prompted for their username and password (unless they specified it using **`--username=`** and **`--password=`** at the command-line).

The Finance Organization Administrator would then be able to see the system from the **Systems** page when logged into the RHN Satellite web interface. The Finance Organization Administrator can then finish the migration process by assigning a base channel and granting entitlements to the client as he would any other system registered to his organization, which is avaiable from the system's **History** page in the **Events** subtab.


RHN Satellite - Systems - Events - History - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Knowledgebase | Documentation
LOGGED IN: admin | Preferences | Sign Out
RED HAT NETWORK SATELLITE
Systems
Search
Overview Systems Errata Channels Configuration Schedule Users Admin Help
No systems selected Manage Clear
Overview
Systems
All
Virtual Systems
Out of Date
Unentitled
Ungrouped
Inactive
Recently Registered
Proxy
System Groups
System Set Manager
Advanced Search
Activation Keys
Stored Profiles
Custom System Info
Kickstart
Event Type
delete system
Details Software Configuration Provisioning Groups Events
Pending History

satellite.example.com

System History

The following history events have been noted for this system.

Please note that this system has no pending events.

1 - 7 of 7

Type	Status	Summary	Time
	(n/a)	added system entitlement	02/24/09 10:59:53 AM EST
	(n/a)	added system entitlement	02/24/09 10:59:30 AM EST
	(n/a)	System migration	02/24/09 10:57:56 AM EST
	(n/a)	System migration	02/24/09 10:54:21 AM EST
	(n/a)	removed system entitlement	02/24/09 10:54:17 AM EST
	(n/a)	added system entitlement	02/19/09 2:24:44 PM EST
	(n/a)	subscribed to channel rhel-i386-server-5	02/19/09 2:24:44 PM EST

1 - 7 of 7

Figure 4.2. System History

Note

The Satellite Administrator can migrate a system from one trusted organization to any other in the trust. However, Organization Administrators can only migrate a system from their own organization to another in the trust.

Satellite Administrators that need to migrate several systems at once can use the **`--csv`** option of **`migrate-system-profile`** to automate the process using a simple comma-separated list of systems to migrate.

A line in the CSV file should contain the ID of the system to be migrated as well as destination organization's ID in the following format:

```
systemId,to-org-id
```

The *`systemId`*, for example could be **`1000010000`**, while the *`to-org-id`* could be **`3`**. An example CSV would look like the following:

```
1000010000,3
1000010020,1
1000010010,4
```

For more information about using **`migrate-system-profile`**, refer to the manual page by typing **`man migrate-system-profile`** or for a basic help screen type **`migrate-system-profile -h`** at the command line.

Part II. Satellite Provisioning

This part describes how to manage, allocate resources, and deploy servers and systems using Satellite, including kickstarting physical and virtual server systems, and provisioning supported systems that have no previous operating system installed (also called *bare metal* provisioning).

Provisioning with Satellite

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 has dramatically boosted the flexibility and power of its signature provisioning capabilities.

This document contains concise details and instructions for use of the kickstart provisioning functionality in Red Hat Network Satellite.

5.1. Requirements

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.

5.2. Definitions and Terms

- Provisioning — The process of configuring a machine (physical 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 the Anaconda installation program that allows you 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 in Satellite 5.3.0 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 *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.

5.3. Provisioning Scenarios Supported

- New Installations — Starting 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 supports KVM, Xen fully-virtualized guests, and Xen para-virtualized guests. Previously only Xen paravirtualized guests were 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 the same Satellite instance

5.4. Overview of Preparing a Satellite for Provisioning

1. Synchronize 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

5.5. 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.

5.5.1. 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*.

5.5.2. 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/
   ```

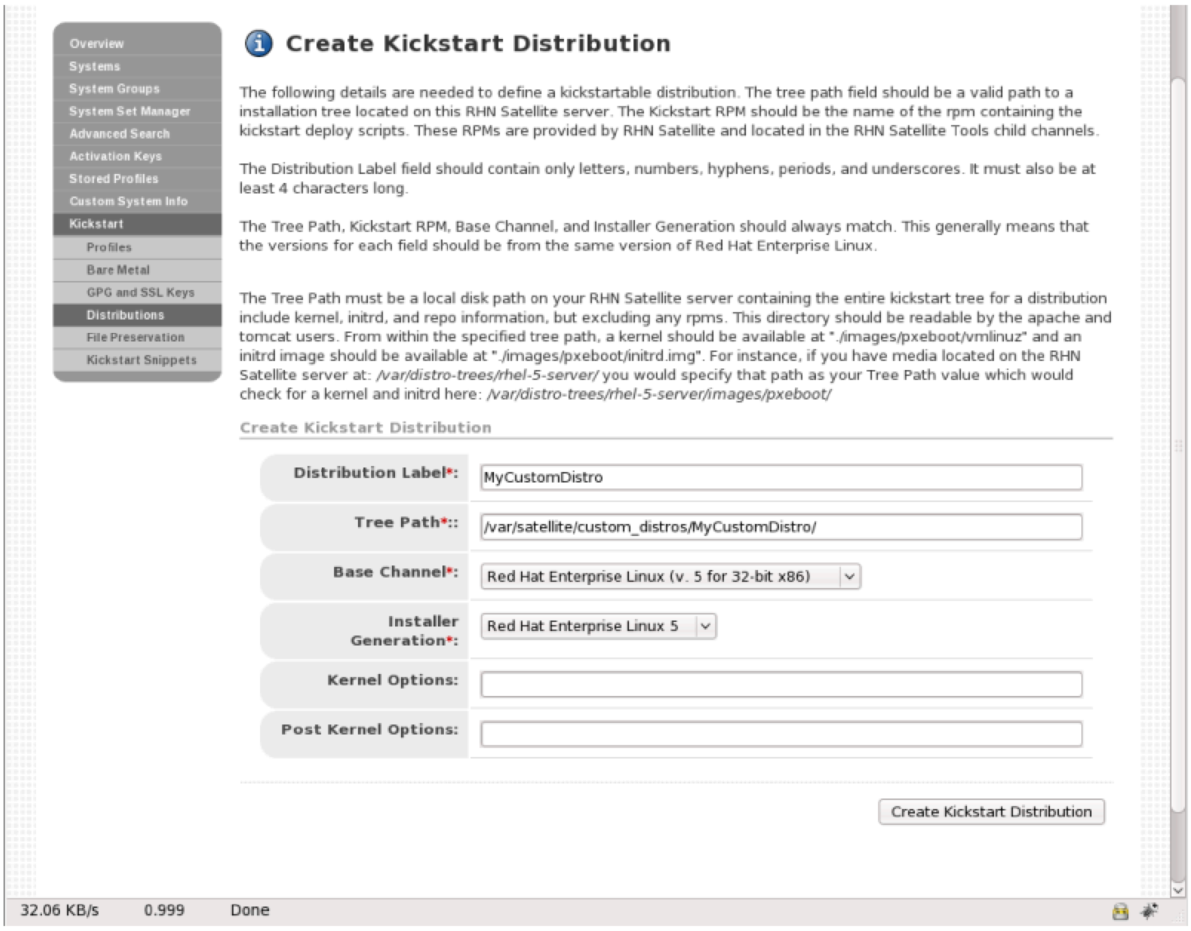
   Note that the sub-directory within the tree may be different depending on your distribution. Once this step is complete, you may delete all of the RPM files from the appropriate directory within the tree path. In this example, run the following:

   ```
   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.



Create Kickstart Distribution

The following details are needed to define a kickstartable distribution. The tree path field should be a valid path to a installation tree located on this RHN Satellite server. The Kickstart RPM should be the name of the rpm containing the kickstart deploy scripts. These RPMs are provided by RHN Satellite and located in the RHN Satellite Tools child channels.

The Distribution Label field should contain only letters, numbers, hyphens, periods, and underscores. It must also be at least 4 characters long.

The Tree Path, Kickstart RPM, Base Channel, and Installer Generation should always match. This generally means that the versions for each field should be from the same version of Red Hat Enterprise Linux.

The Tree Path must be a local disk path on your RHN Satellite server containing the entire kickstart tree for a distribution include kernel, initrd, and repo information, but excluding any rpms. This directory should be readable by the apache and tomcat users. From within the specified tree path, a kernel should be available at "./images/pxeboot/vmlinuz" and an initrd image should be available at "./images/pxeboot/initrd.img". For instance, if you have media located on the RHN Satellite server at: */var/distro-trees/rhel-5-server/* you would specify that path as your Tree Path value which would check for a kernel and initrd here: */var/distro-trees/rhel-5-server/images/pxeboot/*

Create Kickstart Distribution

Distribution Label*:	MyCustomDistro
Tree Path*::	/var/satellite/custom_distros/MyCustomDistro/
Base Channel*:	Red Hat Enterprise Linux (v. 5 for 32-bit x86)
Installer Generation*:	Red Hat Enterprise Linux 5
Kernel Options:	
Post Kernel Options:	

Figure 5.1. Creating Kickstart Distribution

5.5.3. 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.

Architecture	kernel	Initial RAM Disk 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 5.1. Required Distribution Files by Architecture

5.5.4. 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.

5.6. 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.

5.6.1. Virtualization Types

All Kickstart profiles have a virtualization type associated with them:

- None — This profile will be treated as not virtualized at all. Use this profile for normal re-provisioning, bare metal, and non-Xen or KVM installations (such as VMware, Virtage, etc.).
- 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 any hardware support besides a PAE flag on the system CPU, 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 fully-virtualized 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.

5.6.2. 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.

5.6.2.1. 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, which you can view by clicking **Kickstart File**.

5.6.2.2. 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

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:

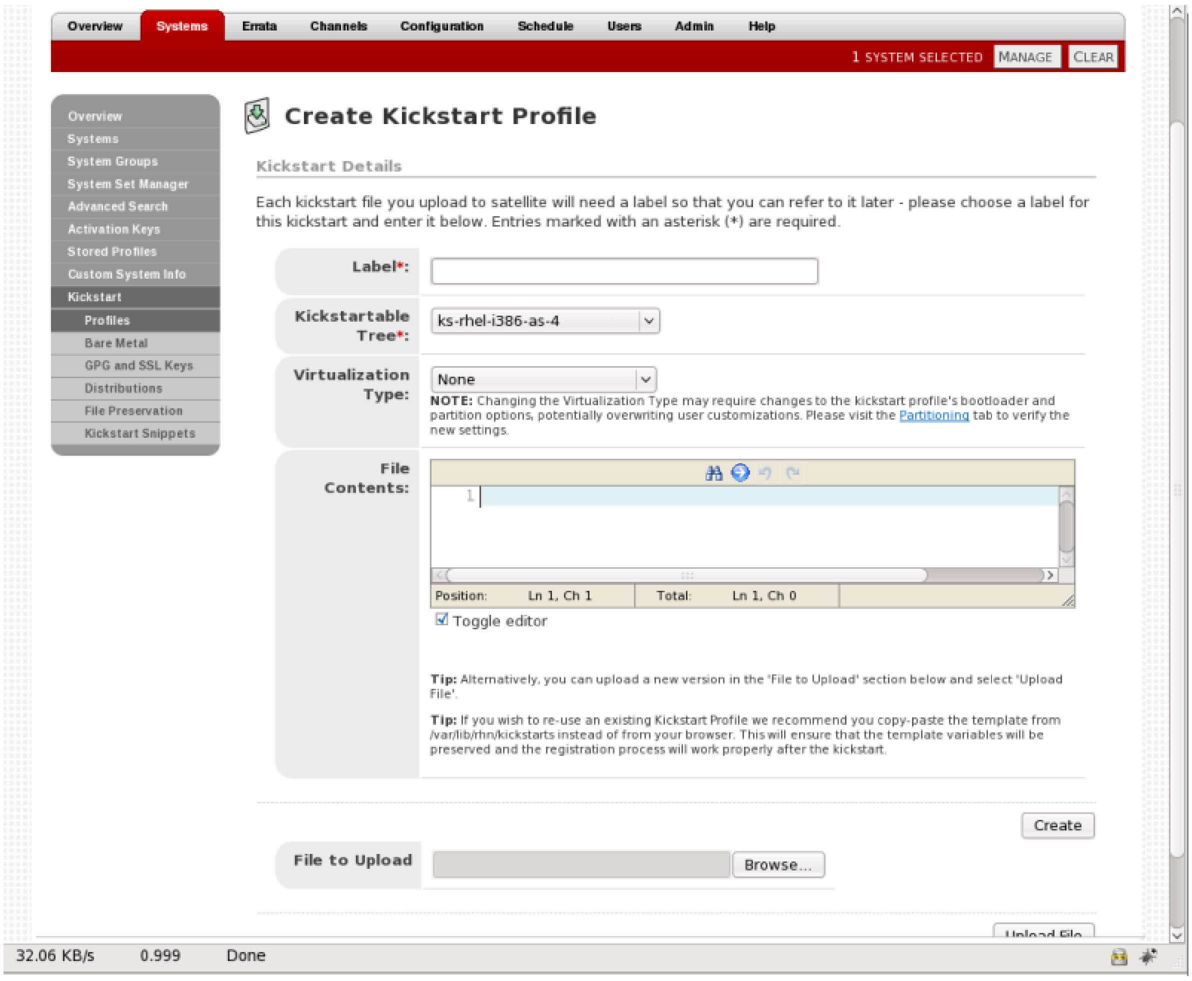
```
install
text
network --bootproto dhcp
url --url http://$http_server/ks/dist/org/1/ks-rhel-i386-server-5
```

```
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 host name. This will be filled in when the kickstart template is rendered. Also the **`redhat_register`** snippet is used to handle registration.



Create Kickstart Profile

Kickstart Details

Each kickstart file you upload to satellite will need a label so that you can refer to it later - please choose a label for this kickstart and enter it below. Entries marked with an asterisk (*) are required.

Label*:

Kickstartable Tree*: ks-rhel-i386-as-4

Virtualization Type: None

NOTE: Changing the Virtualization Type may require changes to the kickstart profile's bootloader and partition options, potentially overwriting user customizations. Please visit the Partitioning tab to verify the new settings.

File Contents:

Position: Ln 1, Ch 1 Total: Ln 1, Ch 0

[✓] Toggle editor

Tip: Alternatively, you can upload a new version in the 'File to Upload' section below and select 'Upload File'.

Tip: If you wish to re-use an existing Kickstart Profile we recommend you copy-paste the template from /var/lib/rhn/kickstarts instead of from your browser. This will ensure that the template variables will be preserved and the registration process will work properly after the kickstart.

File to Upload

Figure 5.2. Raw Kickstart

5.7. Templating

One of the more powerful new features in Satellite 5.3 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.

5.7.1. 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.

5.7.2. 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.

5.7.3. 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.

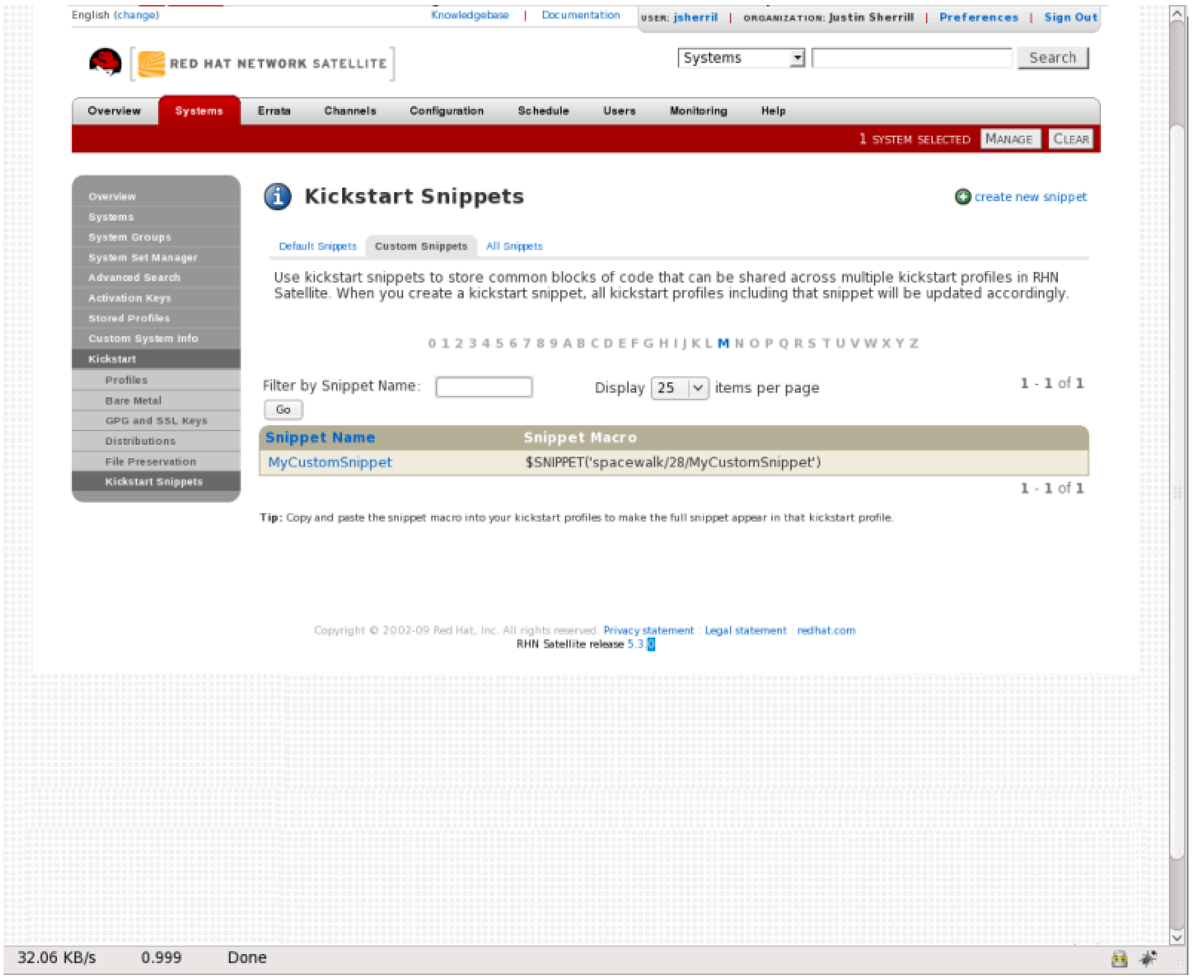
The main purpose of snippets are to be able to reuse pieces of code between multiple kickstart templates and thus make each template easy to understand.

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.



Kickstart Snippets

Use kickstart snippets to store common blocks of code that can be shared across multiple kickstart profiles in RHN Satellite. When you create a kickstart snippet, all kickstart profiles including that snippet will be updated accordingly.

Filter by Snippet Name: Display 25 items per page 1 - 1 of 1

Snippet Name	Snippet Macro
MyCustomSnippet	$SNIPPET('spacewalk/28/MyCustomSnippet')

1 - 1 of 1

Tip: Copy and paste the snippet macro into your kickstart profiles to make the full snippet appear in that kickstart profile.

Copyright © 2002-09 Red Hat, Inc. All rights reserved. Privacy statement : Legal statement : redhat.com
RHN Satellite release 5.3.0

Figure 5.3. Kickstart Snippets

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

5.7.3.1. 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.

5.7.3.2. 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/

5.8. Kickstarting a Machine

5.8.1. 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

5.8.1.1. 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.

5.8.1.2. 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 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.
- You must have the TFTP service on and running.

5.8.1.2.1. Configuring an Existing DHCP Server

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 to edit the DHCP configuration file so that it points to the Cobbler server and PXE boot image.

As root on the DHCP server, edit the /etc/dhcpd.conf file and append a new class with options for performing PXE boot installation. For example:

```
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";
}
```

Following each action step-by-step in the above example:

1. The administrator enables network booting with the **`bootp`** protocol.
2. Then, the administrator creates a class called **`PXE`**, which, if a system that is configured to have PXE first in its boot priority, identifies itself as **`PXEClient`**.
3. Then DHCP server then directs the system to the Cobbler server at 192.168.2.1.

4. Finally, the DHCP server refers to the boot image file (in this case, at **`/var/lib/tftpboot/pxelinux.0`**.

5.8.1.2.2. Xinetd and TFTP

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.

To configure TFTP, you must first enable the service via Xinetd. To do this, edit the **`/etc/xinetd.d/tftp`** as root and change the **`disable = yes`** line to **`disable = no`**.

Alternatively, you can use the following command:

```
chkconfig xinetd on
```

Before TFTP can start serving the **`pxelinux.0`** boot image, you must start the Xinetd service.

```
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.

5.8.1.2.3. Configuring SELinux and IPTables for Cobbler Support

Red Hat Enterprise Linux is installed with SELinux support in addition to secure firewall enabled by default. To properly configure a Red Hat Enterprise Linux server to use Cobbler, you must first configure these system and network safeguards to allow connections to and from the Cobbler Server.

5.8.1.2.3.1. SELinux Configuration

To enable SELinux for Cobbler support, you must set the SELinux boolean to allow HTTPD web service components. Run the following command as root on the Cobbler server:

```
setsebool -P httpd_can_network_connect true
```

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

You must also set SELinux file context rules to ensure Cobbler properly functions in an SELinux system.

Run the following as root on the Cobbler server:

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

The command sets file context for TFTP to serve the boot image file.

5.8.1.2.3.2. IPTables Configuration

Once you have configured SELinux, you must then configure IPTables to allow incoming and outgoing network traffic on the Cobbler server.

If you have an existing firewall ruleset using IPTables, you need to add the following rules to open the requisite Cobbler-related ports. The following lists each of the requisite rules with their associated service.

- 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 tcp -p udp --dport 25150 -j ACCEPT
  ```

- For Koan:

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

Once those firewall rules are entered, be sure to save the firewall configuration:

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

5.8.1.2.4. Syncing and Starting the Cobbler Service

Once all the prerequisites specified in **`cobbler check`** are met, you can now start the Cobbler service.

First, ensure that the configuration files are all synchronized by running the following command:

```
cobbler sync
```

Then, 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.

5.8.1.2.5. 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.

5.8.1.2.6. 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.

5.8.1.3. 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 Satellite 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.

5.8.2. 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 establish 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** and **`spacewalk-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.

5.8.2.1. 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.

5.8.3. 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** system 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 512MB)
- 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 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`**)

5.8.4. 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 installation, you can replace the Satellite's FQDN with that of the Proxy's. For example if the URL to your 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
```

5.9. Advanced Topics

5.9.1. 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:

XML-RPC 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, and 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 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 channel, download kickstart file associated to a profile, manipulate advanced options, manipulate custom options, and add pre/post scripts associated to a kickstart profile.

XML-RPC Namespace	Usage
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, and setup locale information associated to a given kickstart profile.

Table 5.2. XML-RPC Methods

Additionally, the following API methods calls may be used to re-provision a 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.

5.9.2. 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 synchronized 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 shell on host where the satellite is installed:

```
sudo cobbler profile list
```

To get a list of kicktstart trees (and kernels, ramdisks, 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 Web UI, 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
```

5.9.3. 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]
```

5.9.4. 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`**

You will encounter these names if you choose to interact with Cobbler directly at the command line. Note that it is important that you do *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.

5.9.5. 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
```

5.9.6. 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
```

5.10. Troubleshooting

5.10.1. Web Interface errors

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

`/var/log/httpd/error_log` — Check this logfile second for possible sources of web UI errors

5.10.2. 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://satellite.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

5.10.3. 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://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 you should first check that the above URL will generate a 200 response:

```

wget http://satellite.example.com/ks/dist/ks-rhel-i386-server-5-u3
--2009-08-19 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'
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://satellite.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.

5.10.4. 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.

5.10.5. Tracebacks from Taskomatic

If you receive emails such as:

```
Subject: WEB TRACEBACK from satellite.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(CobblerXMLRPCHelper.java:72)
 at com.redhat.rhn.taskomatic.task.CobblerSyncTask.execute(CobblerSyncTask.java:76)
 at
 com.redhat.rhn.taskomatic.task.SingleThreadedTestableTask.execute(SingleThreadedTestableTask.java:54)
 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(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(HttpURLConnection.java:1236)
 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

5.10.6. 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://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
```

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://satellite.example.com/pub/RHN-ORG-TRUSTED-SSL-CERT -O /usr/share/rhn/RHN-ORG-
TRUSTED-SSL-CERT
```

Fetch the SSL certificate 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 certificate strings from the **`rhn-register`** configuration files.

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

Register to the RHN Satellite with 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, 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 four steps from above into the shell prompt 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.

5.10.7. 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
```

Part III. Working with multiple Satellites

This part describes how to work with multiple Satellite instances, including managing Satellite interaction with Red Hat Network Hosted, the remote network from which Satellite receives supported errata and content updates officially from Red Hat, Inc.

Inter-Satellite Sync (ISS)

As an organization grows, so does complexity of the deployed systems. Red Hat Network Satellite deployment can grow with your IT infrastructure. One capability that assists with managing scale and complexity is the deployment of multiple RHN Satellites configured to communicate with each other. Each Satellite manages a segment of an organization's deployment, minimizing the inefficiencies of both massive scaling and dispersed data centers. With multiple Satellites, an organization can infinitely scale upward, allow semi-autonomous management of IT segments, and ensure maximum management performance of systems closest to the managing Satellite while enforcing content standardization and life-cycle best practices.

RHN Satellite 5.3 supports synchronization between two Satellites. This synchronization, also called *Inter-Satellite Sync*, allows administrators to simplify the process of coordinating content from one RHN Satellite source to another or several others.

6.1. Required Environment for ISS

The following are the basic requirements for Inter-Satellite Sync.

- At least two RHN Satellite 5.3 servers
- At least one RHN Satellite populated with at least one channel
- Master RHN Satellite SSL certificate available on each of the slave RHN Satellites for secure connection

6.2. Recommended Models for Inter-Satellite Sync

The Inter-Satellite Sync feature for Satellite provides facilities for synchronizing content between two or more Satellites. The following are some of the more typical uses that show the possibilities of Inter-Satellite Sync and help guide you in determining how to make the most of this feature in your environment.

Note

If you are not sure if the Inter-Satellite Sync feature is right for your organization, please note that you can continue to use RHN Satellite 5.3 in the typical manner. Installing or upgrading to RHN Satellite 5.3 does not require that you make use of this feature.

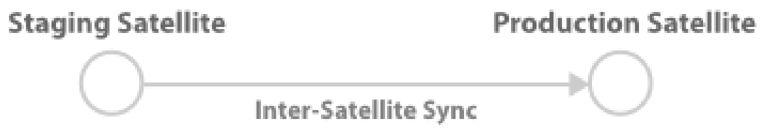


Figure 6.1. Staging Satellite

In this example, the Stage Satellite is used to prepare the content and perform quality assurance (QA) work — to make sure that packages are fit for production use. After content is approved to go to production, the Production Satellite will then synchronize the content from the Stage Satellite.

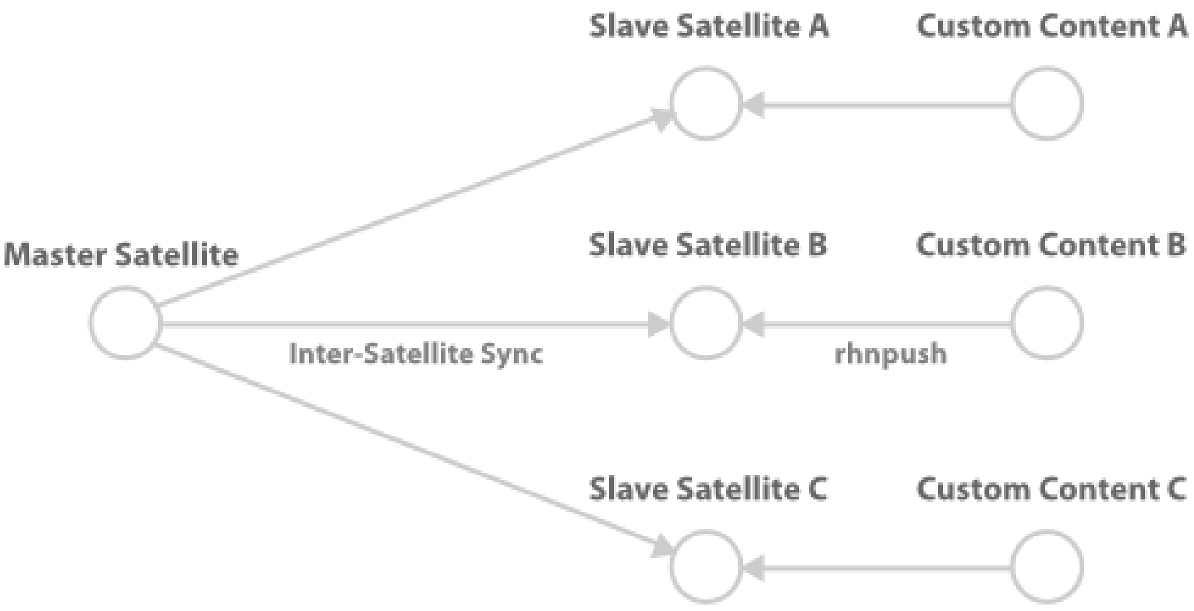


Figure 6.2. Master Server and Slave Peers that include their own custom content

In this example, the master Satellite is the development channel, from which content is distributed to all production slave Satellites. Some slave Satellites have extra content not present in master Satellite channels. These packages are preserved, but all changes from master Satellite are synchronized to Slave Satellite.

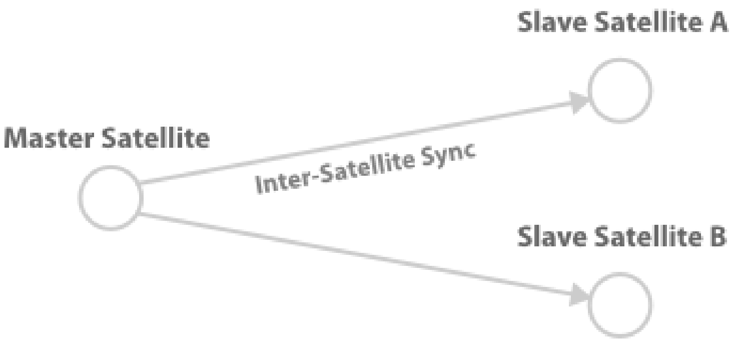


Figure 6.3. Slave Satellites are maintained exactly as the master Satellite

In this example, the master Satellite (for example, a software or Hardware vendor) provides data directly to its customer. These changes are regularly synchronized to slave Satellites.

6.3. Configuring the Master RHN Satellite Server

To use the Inter-Satellite sync feature, you must first ensure that you have it enabled. Make sure that the **`/etc/rhn/rhn.conf`** file contains the following line:

```
disable_iss=0
```

In the same file is the variable:

```
allowed_iss_slaves=
```

By default, no slave Satellites are specified to sync from the master server, so you must enter the hostname of each slave Satellite server, separated by commas. For example:

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

Once you finished configuring the **`rhn.conf`** file, restart the **`httpd`** service by issuing the following command:

```
service httpd restart
```

6.4. Configuring the Slave RHN Satellite Servers

To configure RHN Satellite slave servers for Inter-Satellite Sync, ensure that you have the ORG-SSL certificate from your master RHN Satellite server so you can securely transfer content. This can be downloaded 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 on the slave Satellite, such as the **`/usr/share/rhn/`** directory.

For information about SSL configuration for use with RHN Satellite, refer to Chapter 3, "SSL Infrastructure" in the *RHN Satellite Client Configuration Guide*.

Once the SSL certificate is placed on the slave server, you can see the list of channels available to sync from the master Satellite server by running the following command (replacing the **`master.satellite.example.com`** with the hostname of the master Satellite server):

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

This command lists both Red Hat Official channels as well as any custom channels available on the master Satellite server.

6.5. Using Inter-Satellite Sync

Now that Inter-Satellite Sync is configured, you can now use it to synchronize channels from the master Satellite to the slave servers.

On the slave servers, configure the Master server hostname and SSL certificate file path in the following lines of the **`/etc/rhn/rhn.conf`** file:

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

Then run the **`satellite-sync`** command by typing:

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

Note

Any command line options to the **`satellite-sync`** command will override any default or customized settings in the **`/etc/rhn/rhn.conf`** file.

6.5.1. Syncing between a Development Staging Server and a Production Satellite

There may be instances where an administrator wants to sync data from a staging server that has custom channels that are ready for production use to a production Satellite server.

For example, a production Satellite Server normally syncs directly from RHN Hosted servers for content updates, but will occasionally sync production-ready information from a RHN Satellite development server.

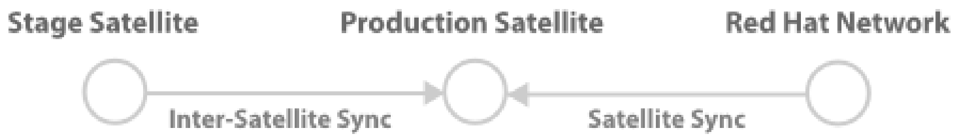


Figure 6.4. Syncing from RHN Hosted and a Satellite Staging Server

Normally, the administrator runs:

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

This command downloads directly from data from **rhn_parent** (usually RHN Hosted, rhn.redhat.com). Then, to sync from the staging Satellite server address , the administrator runs:

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

6.5.2. Bi-directional sync

Administrators can configure an environment where two RHN Satellite servers act as masters of each other. For example, Satellite A and B can sync content from one another.

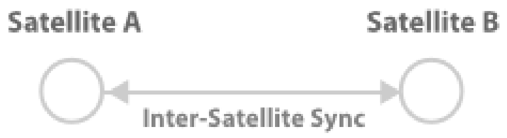


Figure 6.5. Bi-directional syncing

Both Satellites would need to share SSL certificates, then set the **`iss_parent`** option in the **`/etc/rhn/rhn.conf`** file of Satellite A to point to the hostname of Satellite B, and do the same for Satellite B to point to Satellite A as the **`iss_parent`**.

6.6. Synchronizing by Organization

Satellite-sync has a new enhancement as part of the Inter-Satellite sync feature where a user can import content to any specific organization. This can be done locally or by a remote syncing from Hosted or another Satellite.

The aim is for Satellite sync to be able to import content with respect to org_id. This targets two sets of users. One is the disconnected Multi-Org case, where the main source of content for the user is either to get content from channel dumps or to export them from connected satellites and import it to the Satellite. The user mainly hosts custom channels from disconnected satellites. If they wish to export custom channels from connected satellites, they can do so by organizational sync.

The other case is a connected Multi-Org satellite customer. These new flags could work as a means of moving content between multiple orgs.

Synchronizing by organization has a few rules that it follows to maintain the integrity of the source organization.

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

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

1. Import content from master to slave satellite:

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

2. Import content from an exported dump of a specific org:

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

3. Import content from RHN Hosted (assuming the system is registered and activated:

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

Appendix A. Revision History

Revision 1.0	Mon Dec 7 2009

Index