Red Hat Satellite 6.2

Transition Guide

Transitioning from Satellite 5 to Satellite 6

Last Updated: 2018-05-01

Red Hat Satellite 6.2 Transition Guide

Transitioning from Satellite 5 to Satellite 6

Red Hat Satellite Documentation Team
satellite-doc-list@redhat.com

Abstract

This document describes how to prepare an existing Satellite 5.6 or 5.7 Server for transition to a new Satellite 6 Server. It describes the necessary preparations and prerequisites, the tools required, and how to install and use those tools to transition your Satellite 5.6 or 5.7 deployment to a newly-installed Satellite 6 Server.

Table of Contents

CHAPTER 1. INTRODUCTION

This chapter describes different methods of migrating from Red Hat Satellite 5 to Satellite 6.

1.1. TRANSITION PATHS

The two primary strategies for migrating from Red Hat Satellite 5 to Satellite 6 are *passive transition* and *active transition*, both of which involve developing a Satellite 6 architecture alongside your Satellite 5 architecture.

Passive Transition

In a passive transition scenario, the existing workloads remain on Satellite 5, and only new workloads and projects are targeted for management by Satellite 6. This strategy is most appropriate if your Satellite deployment is complex and involves significant integration with other applications via APIs or other processes. In this case, Red Hat Satellite 6 is deployed to manage only new workloads and projects, whereas Red Hat Satellite 5 is maintained in order to manage existing workloads until they are retired. Satellite 5 data models might or might not be transitioned to Satellite 6. A passive transition gives administrators the most freedom to reconsider their infrastructure with the least possibility of any disruption to services.

Active Transition

In an active transition scenario, the intention is that all workloads will be moved to Satellite 6. The originating Satellite 5 is then free to be migrated into an archived state and shut down. There are two variations of this strategy:

- Active Transition with Post-Installation Components
- Active Transition without Post-Installation Components

Post-installation components refers to anything specific to Satellite 5 that you configured or created after installation. For example, system groups, activation keys and channels.

Active Transition with Post-Installation Components

Build a new Satellite 6 based upon Satellite 5, and migrate clients after translating the Satellite 5 data models to Satellite 6 and registering systems to the new Satellite. This method allows data models to remain somewhat similar and familiar, with all workloads migrated to the new Red Hat Satellite. This method uses the *sat5to6* script. See Performing the Transition Using the sat5to6 Script.

Active Transition without Post-Installation Components

Build a new Satellite 6 **not** based upon Satellite 5, but using the new features of Satellite 6. Existing workloads are only migrated after a period of testing and planing has taken place. No clients profiles are migrated in this scenario, advantage is taken of the new features provided by Puppet. Red Hat gives customers a whole year of duplicate Satellite subscriptions so that you can build and test a Satellite 6 Server before migrating **only** the Satellite 5 clients to the new system. This version of the active migration strategy is now strongly recommended. This method uses the bootstrap script to migrate the clients. See Chapter 3, *Migrating Existing Systems Using The Bootstrap Script*.

1.2. FREQUENTLY ASKED QUESTIONS

Can I perform an in-place upgrade from Satellite 5 to Satellite 6?

The underlying technologies between Satellite 5 and 6 are significantly different. For this reason, the in-place upgrade process (such as from version 4.x to 5.x or from 5.x to 5.y) does not apply for version 5 to 6. Satellite 6 needs to be installed on a new server as part of the side-by-side transition process. Consequently, this is referred to as a transition process and not an upgrade process.

Which versions of Satellite 5 are supported for the transition?

The transition path begins with Satellite 5.6 or 5.7. If you are running an earlier version of Satellite you first need to upgrade to at least version 5.6.

Future releases of Satellite will also be supported as platforms for moving from Satellite 5.

NOTE

If you plan a more passive transition process, the prior Satellite version does not matter. A passive transition process is more of an adoption process where legacy workloads remain on Satellite 5 and new workloads are freshly modeled on Satellite 6 as if it is treated as new infrastructure for your Red Hat Enterprise Linux Systems Management needs.

Which versions of Red Hat Enterprise Linux will Satellite 5 and Satellite 6 support as client systems?

The following support matrix currently applies:

- Satellite 5.6 supports clients of Red Hat Enterprise Linux versions 4, 5, 6, and 7.
- Satellite 6 supports clients of Red Hat Enterprise Linux versions 5.7 and later, 6.1 and later, and 7.0 and later.
- Satellite 6 does not support clients of version Red Hat Enterprise Linux 4.

What is the quick version of how to transition from Satellite 5 to Satellite 6?

1. Install Satellite 6, activate with a manifest and synchronize Red Hat content from CDN.
    a. Review the documentation, learn and understand the basics of the Satellite 6 Product, including new concepts, such as environments.
    b. Back up the new Satellite 6 Server, including databases, before using the transition tools.
2. As root on Satellite 5, run the exporter command-line tool.
3. Copy the resulting data onto Satellite 6.
4. As root on Satellite 6, run the importer command-line tools.
5. Validate and test the resulting Satellite 6 system for a subset of end to end functionality and important use cases corresponding to the data types transitioned from Satellite 5 to 6.

NOTE

Not all of the concepts can be translated between Satellite 5 and 6. Some manual steps are required to fully populate the newly installed Satellite 6 Server.

Will the transition tools work with a disconnected server?

Yes, the transition tools are designed such that a disconnected environment is assumed, and neither Satellite 5 nor 6 can directly communicate with each other or the Internet in general.

Will the transition tools transition all data from Satellite 5 to Satellite 6?

No. In some situations, Satellite 5 and Satellite 6 take different approaches to the tasks of synchronizing and provisioning hosts, managing entitlements, and dealing with disconnected environments. Some configuration, system, and host data is also created and stored differently. Consequently, not all data can be successfully transitioned.

In Satellite 6, the following information cannot be transitioned:

- Activation-keys that use "Red Hat default"
- Anything history or audit related (events, oscap runs, and so on)
- Anything monitoring related
- Configuration-channel ordering
- Distribution-channel mapping
- Kickstart data (other than snippets)
- Organization entitlement distribution (users need to create their own manifests)
- Organization-trusts settings
- Snapshots
- Stored package profiles
- Custom system information, such as key-value pairs, system notes, and system properties in general. The latter is completed when a system registers to Satellite 6 and connects to the created profile.
- User preferences

Can I run the transition tools and process multiple times?

Yes, the transition tools are idempotent. You can reuse the tools to create Satellite 6 data and preserve previous information. A small data file on the Satellite 6 Server preserves the history of previous runs and tracks what has already been imported.

Can I delete imported entities?

Yes. You can use the `hammer import` command to delete any or all of the entities imported to Satellite 6 from the Satellite 5 export file. This is useful if the import fails for some reason, and you need to update the CSV file and repeat the import. You can also use this procedure to test your import process in a development environment before using it in production.

You can delete a subset of imported entities, for example, users, or you can delete all imported entities. If you delete all imported entities, you effectively revert your Satellite 6 system to its original state before you started the import process.

The following example describes how to delete all imported users.

Example 1.1. Deleting Imported Users

```
$ hammer import user --csv-file /tmp/exports/users.csv --delete --
verbose
Deleting from /tmp/exports/users.csv
Deleting imported user [1->4].
Deleting imported user [2->5].
Deleting imported user [3->6].
Deleting imported user [5->7].
Deleting imported user [6->8].
Deleting imported user [7->9].
Deleting imported user [4->10].
Summary
Deleted 7 users.
```

To delete all imported entities, use the following command:

```
# hammer import all --delete
```

IMPORTANT

This command deletes all imported entities, including users, organizations, repositories, and so on.

Can I use the export tools on my production Satellite 5 system?

Yes, assuming that you run the tools with sufficient disk space. Disk space will vary, from 20 MB to several Gigabytes, depending on the total amount of non-Red Hat content stored on the Satellite in **/var/satellite/redhat/** and being selected for export by the export tool. When exporting custom or cloned channels, the tool searches for non-Red Hat content associated with the channels, to copy into the export archive. Other than some short-term CPU and memory consumption, the tool only performs read-only file system, and SQL queries on the database to gather data, write the resulting data to files on disk, and create a tar archive of the data.

CHAPTER 2. COMPARING SATELLITE 5 AND SATELLITE 6

2.1. DESIGN AND CONCEPTS

2.1.1. Red Hat Satellite 5

Red Hat Satellite 5 is life cycle management tool that includes the ability to deploy, manage and monitor a large number of systems. Satellite 5 can be set up in a connected or disconnected mode in which Red Hat software is distributed to client systems using the original pooled subscription approach. The pooled subscription concept is similar to the way in which clients consume entitlements from Red Hat Network Classic.

Features and Functionality

The popular functionality of Satellite 5 includes the ability to provision a large number of systems using kickstart files and activation keys to install and configure systems to a predictable state. This provisioning process associates systems to designated organizations, software and configuration channels, as well as placing systems in predefined system groups. The Satellite 5 provisioning functionality enables administrators to provision thousands of systems in a consistent manner.

Another popular feature is the ability to manage software and configuration files across large numbers of systems in local or remote environments after those systems have been provisioned. One of the well understood concepts of managing software and configuration files in Satellite 5 is the concept of channels. All software and configuration is managed and distributed through channels, and any client needing access to software or configuration content needs to be associated with one or more relevant channels. Further, the ability to clone channels enabled administrators to create the much needed development-production environments required by most enterprises.

Red Hat Satellite 5 provides organizations with the benefits of Red Hat Network without the need for public Internet access for servers or client systems. This brings together the tools, services, and information repositories needed to maximize the reliability, security, and performance of your systems.

2.1.2. Red Hat Satellite 6

Red Hat Satellite 6 is the evolution of Red Hat’s life cycle management platform. It provides the capabilities that administrators have come to expect in a tool focused on managing systems and content for a global enterprise. Satellite 6 covers the use cases requested by Satellite 5 customers, but also includes functionality that enables larger scale, federation of content, better control of systems during the provisioning process, and a much more simplified approach to life cycle management. Satellite 6 also further evolves the inherent approach to certificate-based entitlements and integrated subscription management.

2.1.3. Comparison of Concepts

The following table outlines some key concepts and their respective implementation in both Satellite 5 and Satellite 6.

Table 2.1. Comparison of Satellite 5 and Satellite 6 Concepts

Concept Description		Satellite 5	Satellite 6
Open source projects A single project approach versus a modular approach.		Spacewalk	Foreman, Katello, Puppet, Candlepin, and Pulp
Subscription types Pool- or channel-based versus certificate-based. Subscription management has improved over the years from a pool- or channel-based approach to a more specific certificate-based approach. Certificate-based subscription management provides better overall control of subscriptions used by clients.		Entitlements	Subscriptions
Subscription methods (or Satellite subscription consumption). The way that Satellite is enabled to synchronize and distribute Red Hat content. Certificates are activated during installation; manifests are uploaded after installation.		Certificate file	Manifest file
Organization management Both Satellite 5 and 6 have a concept of multiple organizations, but Satellite 6 also includes functionality to include the context of the location.		Organizations	Organizations and Locations

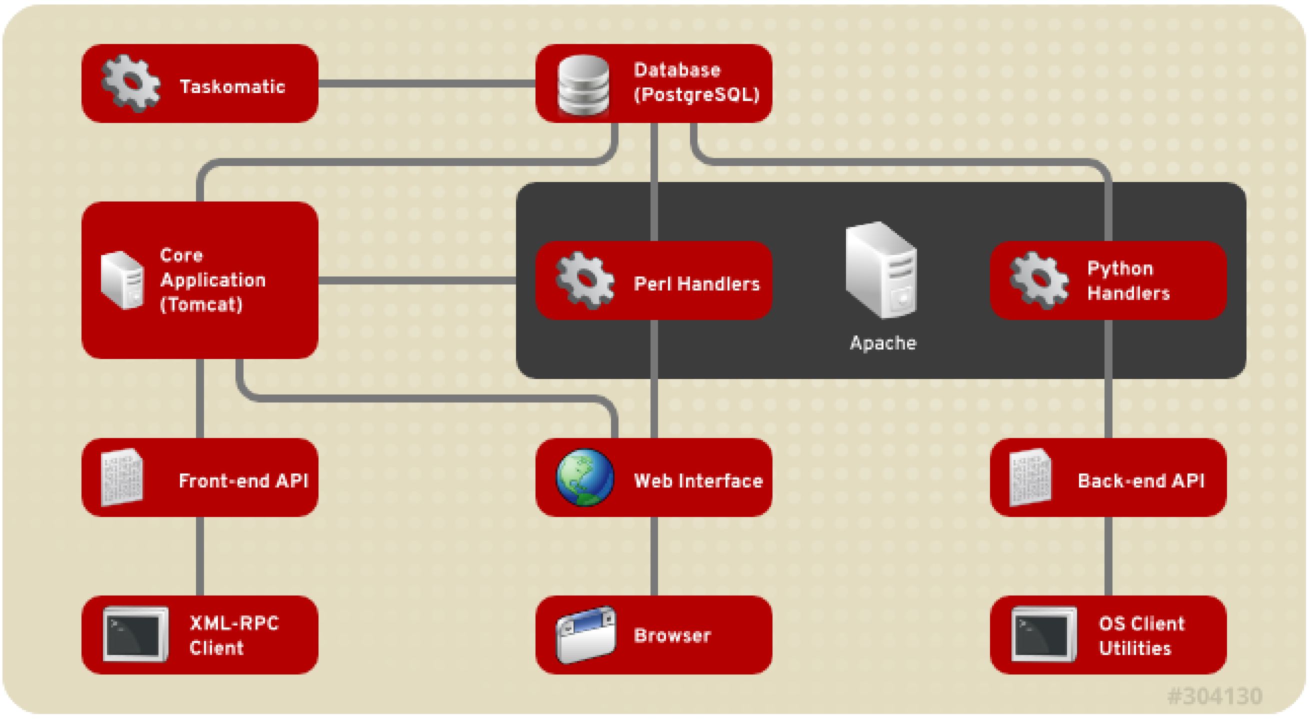
Concept Description		Satellite 5	Satellite 6
Software and configuration content Distributed through channels versus distributed through content views published and promoted through environments. In Satellite 6 a content view contains a chosen set of software repositories and configuration modules that are published and promoted to an environment. Client systems consume its software and configurations through its environment associations.		Software Channels	Products and repositories
Configuration		Configuration Channels	Puppet Repositories
Proxy services		Red Hat Satellite Proxy Server	Red Hat Satellite Capsule Server
Command-line tools		Various CLI tools	Hammer
Virtualization and cloud providers		KVM and Xen	OpenStack, Red Hat Enterprise Virtualizatio n, KVM, VMware, EC2
Database support		Embedded PostgreSQL, managed PostgreSQL, external PostgreSQL, Oracle Database 10g Release 2 or 11g (Standard or Enterprise Edition)	Embedded PostgreSQL for 6.0.

2.2. SYSTEM ARCHITECTURES

2.2.1. Red Hat Satellite 5

Red Hat Satellite 5 is based on an open source project called Spacewalk and is comprised of several key components arranged in the following architecture.

Figure 2.1. Red Hat Satellite 5 System Architecture



Web UI

The Satellite web UI runs through an Apache web server and provides the main entry point for Satellite operations.

Front-end API

The front-end API provides the ability to interact with Satellite 5 through an XML-RPC API. This allows system administrators to write scripts to perform repetitive tasks, or develop third-party applications around Satellite. The front-end API exposes most of the web UI functionality using XML-RPC.

Back-end API

The back end provides a set of APIs that the different client utilities (`rhn_register`, `yum`) connect to. These are not documented and are used solely by the client utilities.

Taskomatic

Taskomatic is a separate service within Red Hat Satellite 5 that runs various asynchronous jobs, such as cleaning up the sessions table, or sending email notifications for new errata. The majority of these jobs run periodically, and you can adjust the frequency with which they occur.

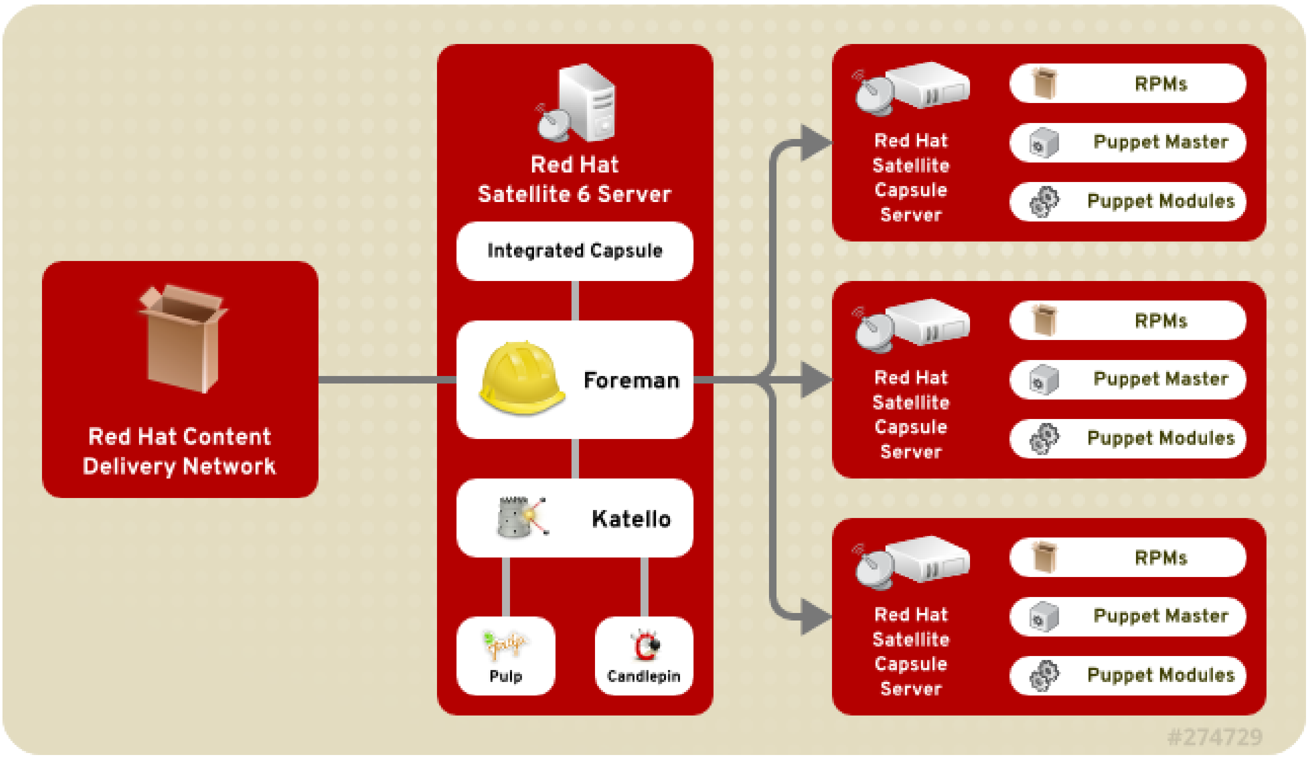
Search Server

Satellite contains a standalone search server that runs as a daemon that allows you to quickly find a system, package, or errata, as opposed to paging through hundreds of items in a list. It uses Apache's Lucene search engine library, which provides more relevant search results and a richer query language.

2.2.2. Red Hat Satellite 6

Red Hat Satellite 6 is based upon several open source projects arranged in the following architecture.

Figure 2.2. Red Hat Satellite 6 System Architecture


Red Hat Content Delivery Network
Red Hat Satellite 6 Server
Integrated Capsule
Foreman
Katello
Pulp
Candlepin
Red Hat Satellite Capsule Server
RPMs
Puppet Master
Puppet Modules
Red Hat Satellite Capsule Server
RPMs
Puppet Master
Puppet Modules
Red Hat Satellite Capsule Server
RPMs
Puppet Master
Puppet Modules
#274729

Foreman

Foreman is an open source application used for provisioning and life cycle management of physical and virtual systems. Foreman automatically configures these systems using various methods, including kickstart and Puppet modules. Foreman also provides historical data for reporting, auditing, and troubleshooting.

Katello

Katello is a subscription and repository management application. It provides a means to subscribe to Red Hat repositories and download content. You can create and manage different versions of this content and apply them to specific systems within user-defined stages of the application life cycle.

Candlepin

Candlepin is a service within Katello that handles subscription management.

Pulp

Pulp is a service within Katello that handles repository and content management.

Hammer

Hammer is a CLI tool that provides command line and shell equivalents of most web UI functions.

REST API

Red Hat Satellite 6 includes a REST-based API service that allows system administrators and developers to write custom scripts and third-party applications that interface with Red Hat Satellite.

Capsule

Red Hat Satellite Capsule Server acts as a proxy for some of the main Satellite functions including repository storage, **DNS**, **DHCP**, and Puppet Master configuration. Each Satellite Server also contains integrated Capsule Server services.

2.3. CONTENT MANAGEMENT

2.3.1. Red Hat Satellite 5

Red Hat Satellite 5 architecture includes Red Hat Satellite Proxy Server, a package-caching mechanism that reduces the bandwidth requirements for Red Hat Satellite and enables custom package deployment. The Satellite Proxy acts as a go-between for client systems and the Satellite Server.

From the client's perspective, there is no difference between a Satellite Proxy and a Satellite. From the Satellite Server's perspective, a Satellite Proxy is a special type of Satellite client.

Satellite Proxy servers are exclusive to Satellite 5; you cannot use Satellite Proxy servers with Satellite 6. Instead, Satellite 6 introduces the concept of Capsules, which provide much the same functionality.

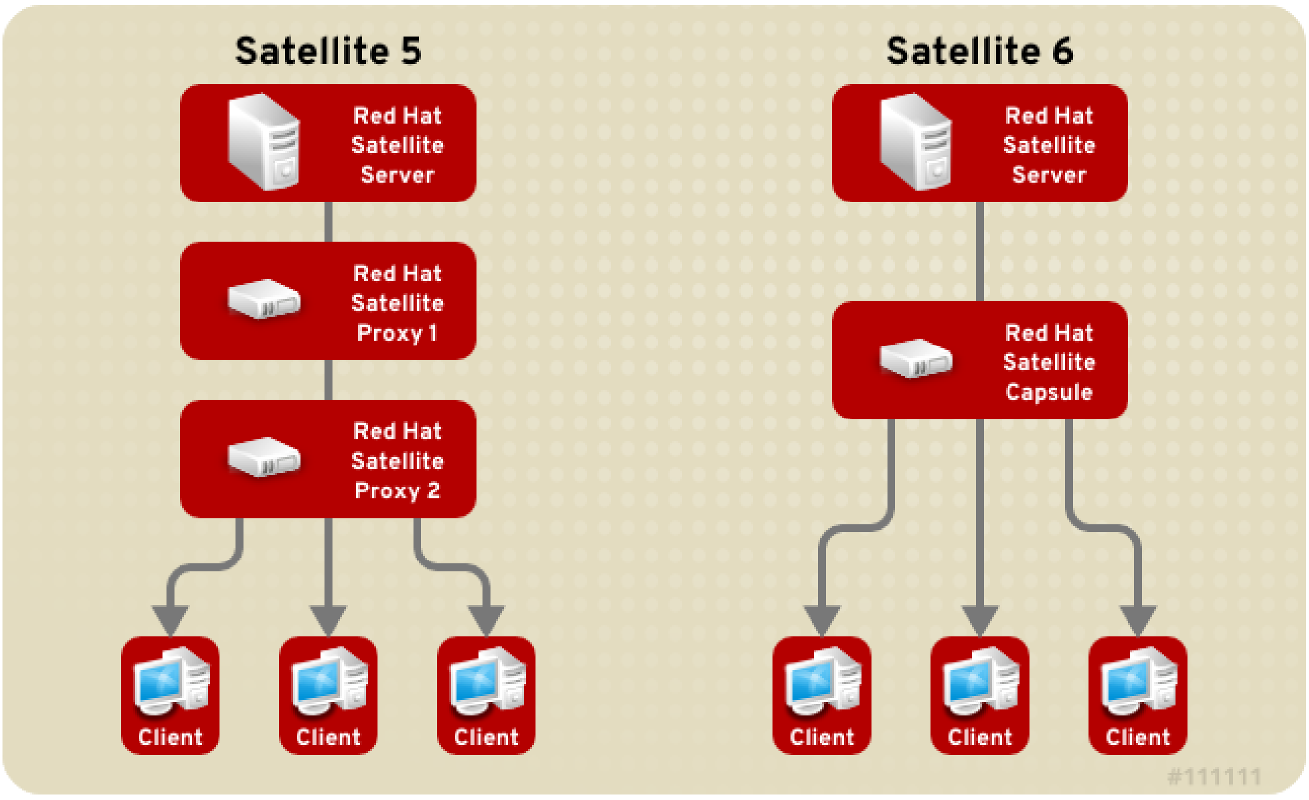
2.3.2. Red Hat Satellite 6

Satellite 6 architecture includes Capsule Servers to provide a similar level of functionality for Satellite 6 that Proxy servers provide for Satellite 5.

IMPORTANT

You cannot tier Capsule Servers the same way you can with Proxy servers. You can see this in the following illustration.

Figure 2.3. Comparison of Satellite 5 Proxy and Satellite 6 Capsule Servers



The first release of Capsule Servers, delivered with Satellite 6.0, can provide the following functionality:

- Mirror repository content (Pulp Node). Content can be staged on a Pulp Node before it is used by a host.
- Mirror Puppet content (Puppet Master)
- Use DHCP, DNS, and TFTP, and integrate with Identity Management (IdM).

2.4. DISCONNECTED CONTENT MANAGEMENT

A key difference between Satellite 5 and Satellite 6 is in the area of "disconnected" content management. Both versions of Satellite can provision and keep hosts synchronized without direct connection to the Internet, but the way they achieve this is slightly different.

2.4.1. Red Hat Satellite 5

Red Hat Satellite 5 achieves disconnected content management by using the **`katello-disconnected`** utility and a synchronization host. An intermediary system with an Internet connection is needed to act as a synchronization host. This synchronization host is in a separate network from Satellite Server.

The synchronization host imports content from the Red Hat Content Delivery Network (CDN). The content is then exported onto a media, such as DVDs, CDs, or external hard drives and transferred to the disconnected Satellite Server.

2.4.2. Red Hat Satellite 6

Red Hat Satellite 6 achieves disconnected content management by using a second Internet facing Satellite and the Inter-Satellite Synchronization (ISS) feature.

The connected Satellite imports content from the Red Hat Content Delivery Network (CDN). The content is then exported onto a media, such as DVDs, CDs, or external hard drives and transferred to the disconnected Satellite Server. The Inter-Satellite Synchronization feature enables full or incremental updates to be made. See Synchronizing Content Between Satellite Servers in the *Content Management Guide* for more information.

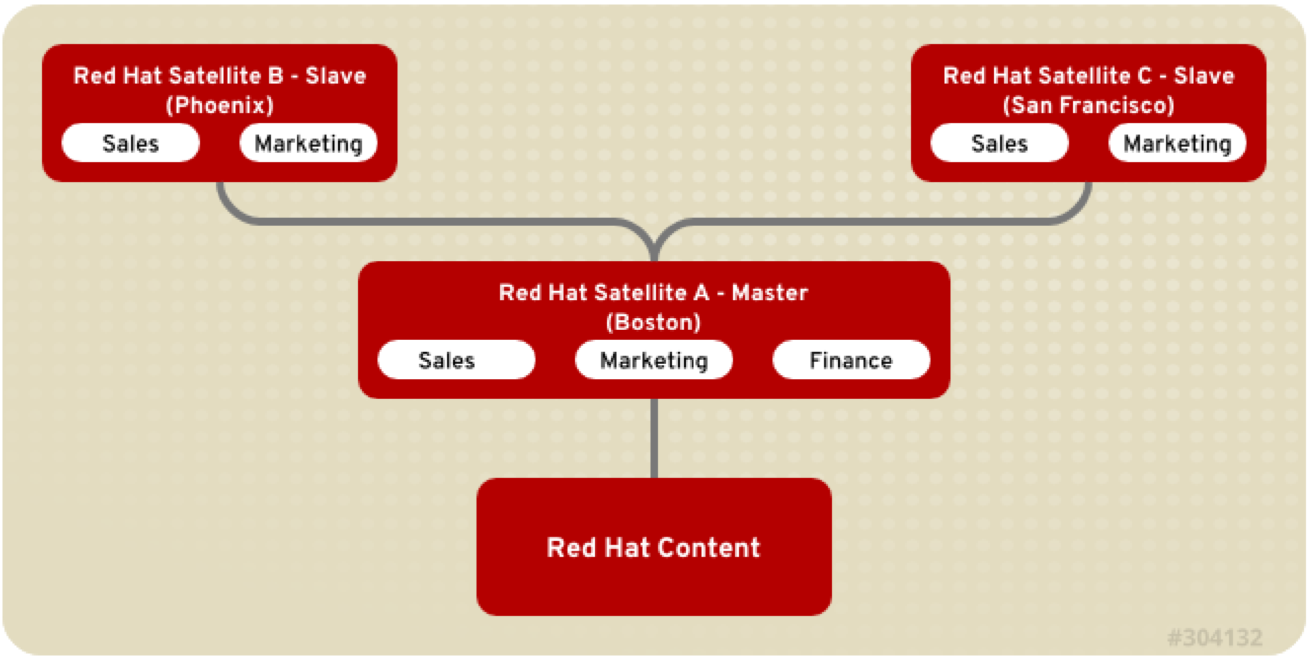
2.5. ORGANIZATIONAL STRUCTURES

2.5.1. Red Hat Satellite 5

Red Hat Satellite 5 can group systems, content, and configuration into multiple and distinct organizations. This is useful for companies with multiple divisions, such a Finance, Marketing, Sales, and so forth. Each organization acts as an individually managed Satellite, each with their own configuration and system profiles. However, Satellite can share content (software channels) among multiple organizations.

Red Hat Satellite 5 also has the ability to synchronize content across multiple Satellite Servers. This allows administrators to provide geographically dispersed Satellites that share the same software channels. For example, a company might have offices in three separate locations and each might require a Satellite, but synchronizing content from a chosen parent Satellite Server.

Figure 2.4. Example Topology for Red Hat Satellite 5

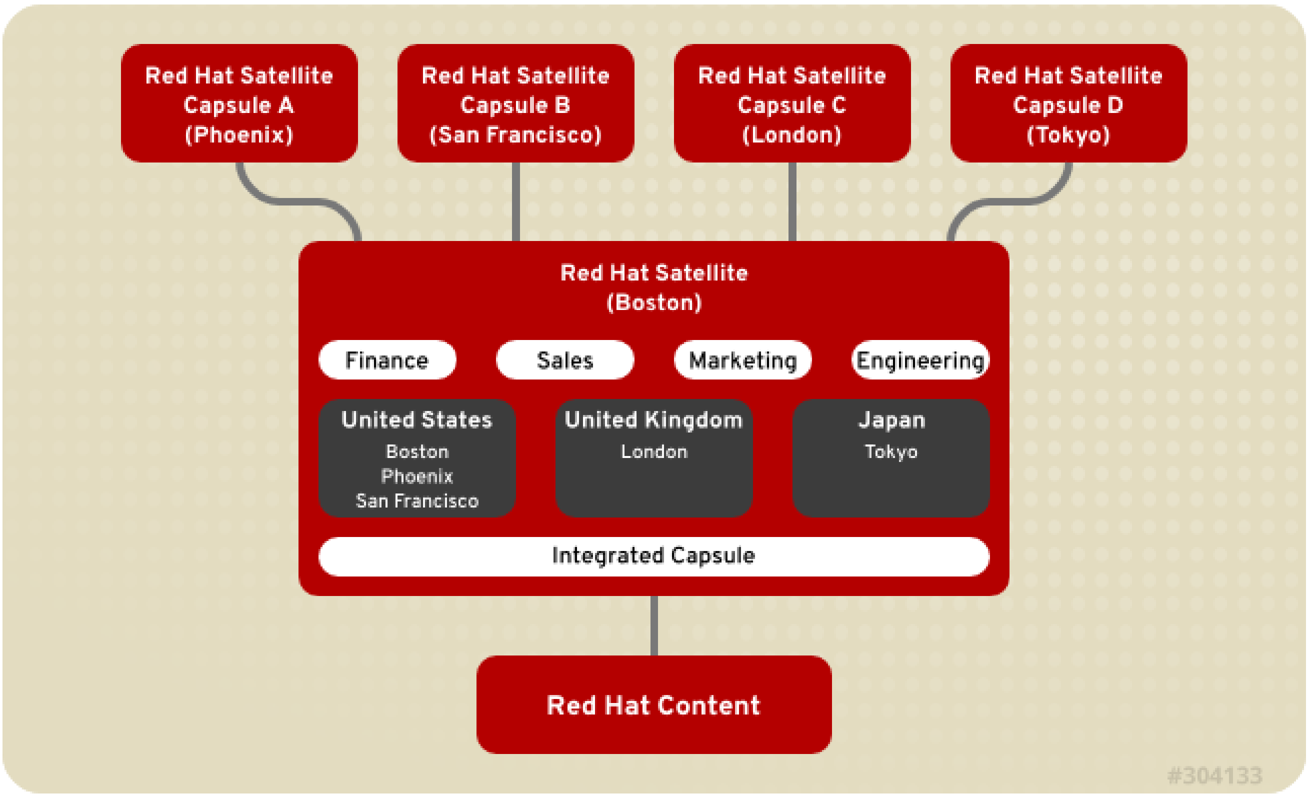

Red Hat Satellite B - Slave (Phoenix)
Sales
Marketing
Red Hat Satellite C - Slave (San Francisco)
Sales
Marketing
Red Hat Satellite A - Master (Boston)
Sales
Marketing
Finance
Red Hat Content
#304132

All systems management operations occur on the Satellite to which they are registered.

2.5.2. Red Hat Satellite 6

Red Hat Satellite 6 takes a consolidated approach to Organization and Location management. System administrators define multiple Organizations and multiple Locations in a single Satellite Server. For example, a company might have three Organizations (Finance, Marketing, and Sales) across three countries (United States, United Kingdom, and Japan). In this example, the Satellite Server manages all Organizations across all geographical Locations, creating nine distinct contexts for managing systems. In addition, users can define specific locations and nest them to create a hierarchy. For example, Satellite administrators might divide the United States into specific cities, such as Boston, Phoenix, or San Francisco.

Figure 2.5. Example Topology for Red Hat Satellite 6


Red Hat Satellite Capsule A (Phoenix)
Red Hat Satellite Capsule B (San Francisco)
Red Hat Satellite Capsule C (London)
Red Hat Satellite Capsule D (Tokyo)
Red Hat Satellite (Boston)
Finance
Sales
Marketing
Engineering
United States
Boston
Phoenix
San Francisco
United Kingdom
London
Japan
Tokyo
Integrated Capsule
Red Hat Content
#304133

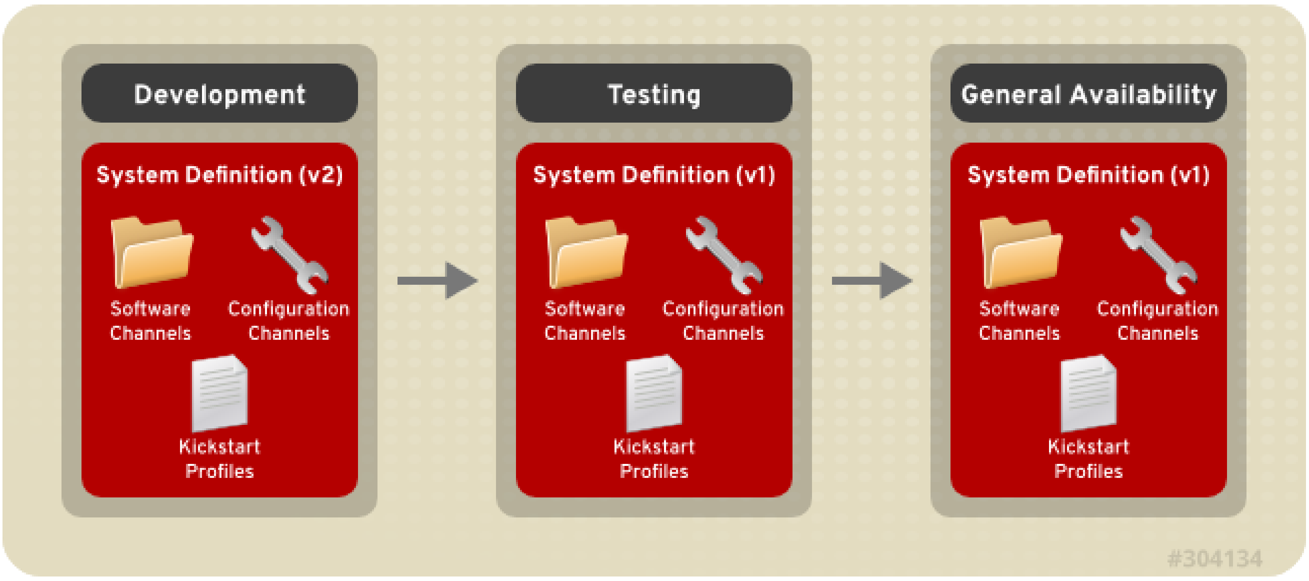
The main Satellite Server retains the management function, while the content and configuration is synchronized between the main Satellite Server and a Satellite Capsule assigned to certain locations.

2.6. APPLICATION LIFE CYCLES

2.6.1. Red Hat Satellite 5

The application life cycle in Red Hat Satellite 5 follows stages in a path. For example, applications might move along a path from "Development" to "Testing" to "General Availability". Each stage in Red Hat Satellite 5 uses System Definitions to manage systems at a particular point in the life cycle. A System Definition in Red Hat Satellite 5 is a set of Software Channels and Configuration Channels that are used in Kickstart Profiles.

Figure 2.6. The Application Life Cycle of Red Hat Satellite 5

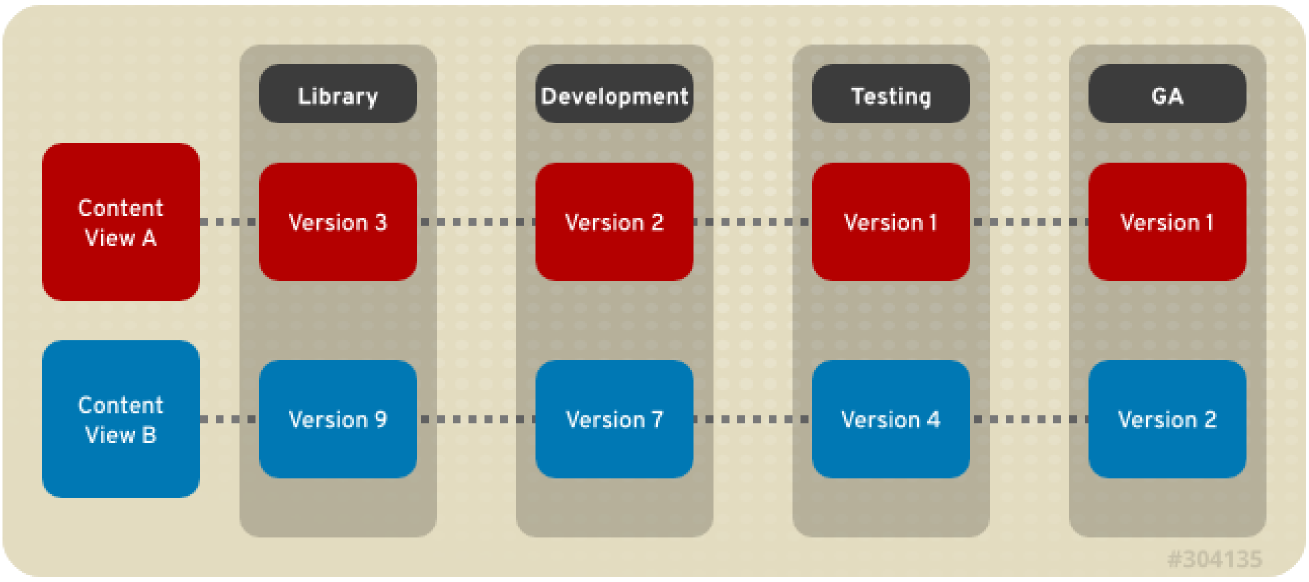


2.6.2. Red Hat Satellite 6

The Red Hat Satellite 6 application life cycle is broken up into two key components: Life Cycle Environments and Content Views.

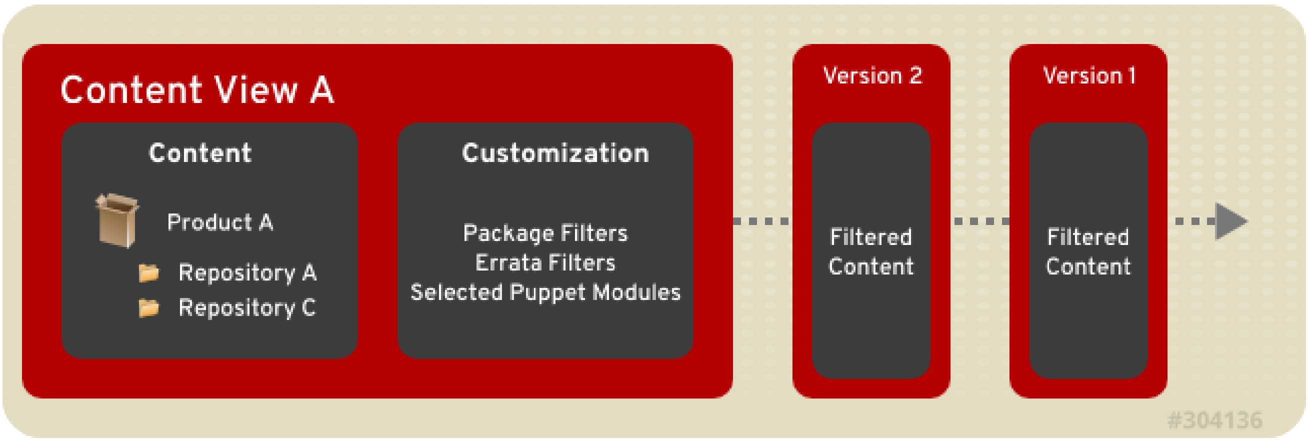
The application life cycle is divided into *life cycle environments*, which mimic each stage of the life cycle. These life cycle environments are linked in an *environment path*. You can promote content along the environment path to the next life cycle stage when required. For example, if development completes on a particular version of an application, you can promote this version to the testing environment and start development on the next version.

Figure 2.7. An Environment Path Containing Four Environments.



Content views are managed selections of content, which contain one or more yum or Puppet repositories with optional filtering. These filters can be either inclusive or exclusive, and tailor a system view of content for life-cycle management. They are used to customize content to be made available to client systems.

Figure 2.8. Creating New Versions of a Content View



Published content views are used with life cycle environments.

CHAPTER 3. MIGRATING EXISTING SYSTEMS USING THE BOOTSTRAP SCRIPT

When the transition tooling is not being used to model a Satellite 6 Server based on Satellite 5, a method is needed to migrate existing clients which also supports adding a Puppet configuration to the clients. Clients can be migrated to Satellite 6 using the following bootstrap script method.

The bootstrap script handles content registration, Product certificates, **and** Puppet configuration. The method using the **`rhn-migrate-classic-to-rhsm`** script does not configure Puppet, which is required for a system in Satellite 6 to be properly subscribed. In addition, the bootstrap script has the advantage of taking a Red Hat Enterprise Linux system, regardless of **where** it is registered (RHN, Satellite 5, SAM, RHSM), or if it is registered at all, and properly subscribe it to Satellite 6.

The bootstrap script package, **`katello-client-bootstrap`**, is installed by default on the Satellite Server's base system and the script itself is installed in the **/var/www/html/pub/** directory to make it available to clients. It can be accessed using a URL in the following form:

```
satellite6.example.com/pub/bootstrap.py
```

The script includes documentation in a readme file. To view the file:

```
$ less /usr/share/doc/katello-client-bootstrap-version/README.md
```

Installing the Bootstrap Script on a Client

As the script is only required once, and only for the **`root`** user, you can place it in **`/root`** and remove it after use, or place it in **`/usr/local/sbin`**. This example will use **`/root`**.

As **`root`**, install the bootstrap script on client systems as follows:

1. Ensure you are in the correct directory. For example, to change to **`/root`**:

   ```
   # cd
   ```

2. Download the script:

   ```
   # wget http://satellite6.example.com/pub/bootstrap.py
   ```

   This will install the script to the current directory.

3. Make the script executable:

   ```
   # chmod +x bootstrap.py
   ```

4. To confirm that the script can now be run, view the usage statement as follows:

   ```
   # ./bootstrap.py -h
   ```

5. Optionally, when the transition process is complete, remove the script:

   ```
   # cd
   # rm bootstrap.py
   ```

Migrating a Red Hat Enterprise Linux 6 System

Prerequisites:

- The bootstrap script is installed as described previously;
- You have an activation key from the Satellite 6 Server. See Section 4.6.5, “Importing Activation Keys”.

1. Enter the bootstrap command as follows with values suitable for your environment.
   For the `-s, --server` option specify the FQDN name of the Capsule; in this example the integrated Capsule is being used so the URL is the FQDN of the Satellite. For options such as location, organization, and host group, use quoted names, not labels, as arguments to the options:

   ```
   # bootstrap.py --login=admin \
   --server satellite6.example.com \
   --location="Example Location" \
   --organization="Example Organization" \
   --hostgroup="Example Host Group" \
   --activationkey=activation_key
   ```

   The script will prompt you for the password corresponding to the Satellite user name you entered with the `-l, --login` option. As a result, it will run and send notices of progress to **stdout**. Watch for output prompting you to approve the certificate. For example:

   ```
   [NOTIFICATION], [2016-04-26 10:16:00], [Visit the UI and approve
   this certificate via Infrastructure->Capsules]
   [NOTIFICATION], [2016-04-26 10:16:00], [if auto-signing is
   disabled]
   [RUNNING], [2016-04-26 10:16:00], [/usr/bin/puppet agent --test -
   -noop --tags no_such_tag --waitforcert 10]
   ```

   The client will wait indefinitely until an administrator approves the Puppet certificate.

2. Sign the Pupper certificate as follows:
   a. In the web UI, navigate to **Infrastructure** > **Capsules**.
   b. Select **Certificates** to the right of the name of the Capsule corresponding to the FQDN given with the `-s, --server` option.
   c. In the **Actions** column select **Sign** to approve the client’s Puppet certificate.
   d. Return to the guest to see the remainder of the bootstrap process completing.

3. In the web UI, navigate to **Hosts** > **All hosts** and ensure that the client is connected to the correct host group.

CHAPTER 4. TRANSITIONING FROM SATELLITE 5 TO 6

This chapter describes the prerequisites for performing a successful transition from Red Hat Satellite 5 to Satellite 6, and the basic transition workflow. It also describes how to install and use the required transition tools, and covers the most common transition use cases.

4.1. PREREQUISITES AND ASSUMPTIONS

The following list describes the prerequisites and assumptions you need to address to ensure a successful transition from Red Hat Satellite 5.6 or 5.7 to Satellite 6.

- Ensure you start with a fully-functional, up-to-date Satellite 5.6 or 5.7 Server.
- Satellite 6 must be correctly installed and configured on a second machine. This installation must include a manifest that provides access to the same content as the Satellite 5.6 or 5.7 instance, and for the same number of clients.
- The Satellite 6 instance must be synchronized with the desired Red Hat content before you start the transition process. The transition tools are purposely designed not to copy Red Hat content from Satellite 5 to Satellite 6.
- The migration tools are designed to work in environments where network security prevents direct connection between the Satellite 5 and Satellite 6 instances. Consequently, the tools were designed to work in isolation, in two stages. The first stage is to export data from Satellite 5, and the second stage is to import data into Satellite 6. After the first stage is complete, you can use network tools (for example, **`SCP`**) or physical media to transfer the exported data to the Satellite 6 Server's base system.
- The export tools run on the Satellite 5 system. This allows direct access to the Satellite 5 database, repositories, and existing tools such as **`spacewalk-report`**. It is also necessary because some customer sites have networking restrictions that limit access to the Satellite 5 instance.
- The import tools run on the Satellite 6 system, for similar reasons.

Given the differences in data models and functionality between Satellite 5 and Satellite 6, importing Satellite 5 data cannot achieve a perfectly-configured Satellite 6 instance. The goal is to reduce the amount of configuration as much as possible. The main purpose of the transition tools is to create an initial setup of a new Satellite 6 installation. It is possible to import data from your Satellite 5 Server at any time, and the transition tools will attempt to handle any data conflicts that occur. However, Red Hat recommends that you perform the transition before you perform any tasks except required configuration on your Satellite 6 Server.

4.2. THE TRANSITIONING WORKFLOW

The transitioning workflow consists of two main phases, each consisting of several steps. These are described in the following sections.

Preparing for Transition

The preparation phase consists of the following steps:

1. Install Satellite 6.

2. Navigate to the Red Hat Customer Portal and create a manifest for each organization that you are going to import.

3. Download those manifests to the Satellite 6 base system. Rename each manifest according to the organization for which it is intended. Replace each and every punctuation mark or space with an underscore. For example, an organization called `MyOrg, Inc.` can use the **MYORG__INC_.zip** manifest.

4. Use the `hammer import organization` command to import your organizations and their manifests. For example:

```
# hammer import organization --csv-file /tmp/exports/users.csv \
--upload-manifests-from /root/manifest-dir
```

Performing the Transition

After you have completed the steps described in Section 4.2, "The Transitioning Workflow" , the Satellite 6 instance is ready to accept transition data. This phase consists of the following steps:

1. Export the entities (metadata and content) from an installed Satellite 5 system, using tools installed on the Satellite 5 Server machine.

2. Transfer the exported entities to the Satellite 6 instance, using **SCP**, a USB device, or other suitable means.

3. Import the Satellite 5 entities, using tools installed on the Satellite 6 Server machine.

4. Tune the Satellite 6 configuration to suit your deployment.

The existing `spacewalk-report` tool extracts most of the information from the Satellite 5 instance. Several new reports have been added, as well as two new tools that are used as part of the export process:

- `spacewalk-channel-export`: This tool exports channel data and metadata beyond that available at the report level.

- `spacewalk-export`: This tool is a wrapper that combines the various export processes into a single tool, and packages the resulting export data into a single data set that can be transferred to the Satellite 6 system.

The existing `hammer` tool has been extended with a new plug-in. This plug-in is compatible with the Satellite 5 export data format, and accesses the Satellite 6 instance using its public API. The import tools track which Satellite 5 entities have been mapped to which Satellite 6 entities. This means you can run the import tools multiple times without duplicating entities.

4.3. PREPARING FOR THE TRANSITION

The transition tools and process assume the following conditions are true:

- A fully-functional Satellite 5.6 or 5.7 instance exists.

- Satellite 6 has been correctly installed on a new machine.

- Direct connectivity between the Satellite 5 and Satellite 6 instances cannot be guaranteed.

- The Satellite 6 instance contains a manifest that enables access to the same content that the Satellite 5 instance accesses, for the same number of client machines.
- The Satellite 6 instance has already synchronized the desired Red Hat content. The transition tools make every attempt not to copy Red Hat content from Satellite 5 to Satellite 6.
- The export tools execute on the Satellite 5 system. This is necessary to allow direct access to the Satellite 5 database, repositories, and existing tools such as **`spacewalk-report`**.
- The import tools execute on the Satellite 6 system, for similar reasons.
- The result of importing Satellite 5 data is **not** a perfectly-configured Satellite 6 instance. The goal is to alleviate as much setup as reasonably possible, given the differences in data models and functionality between Satellite 5 and 6.
- For hosts to be able to check in with Satellite 6, the daemon supplied by the subscription-manager package, **`rhsmcertd`**, must be running. Subscription-manager is installed and enabled by default on Red Hat Enterprise Linux 7, but for Red Hat Enterprise Linux 5 and 6 you must ensure that it is.
  On the hosts, check the status of **`rhsmcertd`**.

  ```
  # service rhsmcertd status
  ```

  If necessary, start **`rhsmcertd`**.

  ```
  # service rhsmcertd start
  ```

  To enable subscription-manager to start at system start.

  ```
  # chkconfig rhsmcertd on
  ```

4.3.1. Preparing the Satellite 5 Server for Transition

Preparing your Satellite 5 Server for transition requires that you subscribe to some specific channels and download extra packages to ensure that your transition runs smoothly. This is described in the following sections.

Installing the Latest Reporting Packages

On Satellite 5, install the latest **`spacewalk-reports`** and **`spacewalk-utils`** packages from the Satellite 5.6 or 5.7 channel.

```
# yum install spacewalk-reports spacewalk-utils
```

The latest **`spacewalk-report`** package provides additional reports which are used by the transition tools:

- activation-keys
- channels
- config-files-latest
- kickstart-scripts

- repositories
- system-profiles

You can use the **spacewalk-report** command to confirm the addition of these new reports.

The **spacewalk-utils** package provides two additional command-line tools, which are used by the transition tools: **/usr/bin/spacewalk-export** and **/usr/bin/spacewalk-export-channels**.

NOTE

The **spacewalk-reports** package is fully supported by Red Hat and provides additional reports designed for use by the **spacewalk-export** tools. You can also use these reports for general, day-to-day reporting.

Installing Extra Transition Packages

Ensure your Satellite 5 Server is subscribed to the RHN Tools channel. This channel provides the latest updates to the **subscription-manager** and **python-rhsm** packages, which are required to install the **subscription-manager-migration** command.

Install the **subscription-manager** and **python-rhsm** packages:

```
# yum install subscription-manager python-rhsm
```

4.3.2. Preparing the Satellite 6 Server for Transition

To prepare the Satellite 6 Server for transition, ensure the **rubygem-hammer_cli_import** package is installed. If necessary, enter **yum install rubygem-hammer_cli_import** to install it from the Satellite 6 repositories.

On Satellite 6, enter the following command to verify that the transition tools were successfully installed:

```
# hammer import --help
Usage:
hammer import [OPTIONS] SUBCOMMAND [ARG] ...
```

4.4. EXPORTING ENTITIES FROM SATELLITE 5

Before you can transition your Satellite 5 data to Satellite 6, you need to export the required entities into a specially-formatted file suitable for use by the Satellite 6 import tools.

The expected workflow on the Satellite 5 Server is to use the **spacewalk-export** command as a wrapper. This wrapper command calls the following commands to export Satellite 5 entities:

spacewalk-report channels

Export all custom and cloned channels and repositories for all organizations.

spacewalk-report activation-keys

Export activation keys.

spacewalk-report kickstart-scripts

Exports kickstart scripts for all organizations.

spacewalk-report users

Export organizations and users.

spacewalk-report system-groups

Export all system groups for all organizations.

spacewalk-report config-files-latest

Export information on configuration channels and the latest configuration file versions.

spacewalk-report repositories

Export repositories.

spacewalk-report system-profiles

Export information about the systems managed by Satellite 5.

Running the exports directly on the Satellite 5 Server allows complete access to the **spacewalk-report** functionality, including limiting the data exported, using the **--where** and **--like** options. However, if the goal is to export as much as possible, or at most limit by organization, the **spacewalk-export** tool can manage the process for you.

The **spacewalk-export** command also uses the **spacewalk-export-channels** command to collect information and content for non-Red Hat channels.

4.4.1. Exporting Data from Satellite 5

On Satellite 5, enter the following command to list the entities that you can export:

```
# spacewalk-export --list-entities

INFO: Currently-supported entities include:
INFO:             channels : Custom/cloned channels and repositories for
all organizations
INFO:      activation-keys : Activation keys
INFO:    kickstart-scripts : Kickstart scripts for all organizations
INFO:                users : Users and Organizations
INFO:        system-groups : System-groups for all organizations
INFO:  config-files-latest : Latest revision of all configuration files
INFO:         repositories : Defined repositories
INFO:      system-profiles : System profiles for all organizations
```

You can use the **--entities** option to limit the export by entity.

```
# spacewalk-export --entities users,repositories
```

You can also use the **--entities** option with the **channels** parameter to export all channel data available on the Satellite 5 instance. Using this format calls both **spacewalk-report channels** and **spacewalk-export-channels** commands, and consequently exports both Red Hat and non-Red Hat channels.

You can use the `--org` option to limit the export by organization. Use the `spacewalk-report users` command to retrieve a list of organization IDs.

```
# spacewalk-export --org=organization-id
```

If you omit the `--org` option, the `spacewalk-export` command will export all channels, including any not assigned to an organization. If you have only one organization then it is recommended to omit the option.

By default, the `spacewalk-export` command stores all exports in the **~/spacewalk-export-dir/exports** file, and packages all export data into the **~/spacewalk-export-dir/spacewalk_export.tar.gz** file. You can use the following options on the command line to specify different values:

```
# spacewalk-export --export-dir=your-export-directory
# spacewalk-export --export-package=your-export-package-name
```

The following is an example of a typical export session:

Example 4.1. Example of a Typical Export Session

```
# spacewalk-export

INFO: Processing channels...

Processing organization: GLOBAL SUPPORT SERVI RED HAT, INC.

* channel: clone-rhel-x86_64-server-5 with: 15778 packages
* channel: clone-rhel-x86_64-server-6 with: 12157 packages
* channel: clone-rhel-x86_64-server-optional-6 with: 6931 packages
.
.
* channel: epel-puppet-rhel6-server-x86_64 with: 8 packages
* channel: puppet-rhel5-server-x86_64 with: 409 packages
* channel: puppet-rhel6-server-x86_64 with: 373 packages

INFO: Processing system-groups...
INFO: Processing activation-keys...
INFO: Processing repositories...
INFO: Processing users...
INFO: Export-file created at /root/spacewalk-export-
dir/spacewalk_export.tar.gz
```

4.4.2. Transferring Exports to Satellite 6

After you have successfully exported all the required entities from your Satellite 5 Server, transfer the **/root/spacewalk-export-dir/spacewalk_export.tar.gz** file to the Satellite 6 Server. If the two servers are connected over the network, you can use **scp** or a similar tool to transfer the file. Alternatively, use removable media such as a USB device or DVD.

WARNING

Red Hat strongly recommends that you place the **spacewalk_export.tar.gz** file in the **/tmp/** directory on your Satellite 6 system. This ensures that the extracted files have suitable permissions for the import process.

Extracting the Exported Archive

Extract the **spacewalk_exports.tar.gz** archive into the **/tmp/** directory on your Satellite 6 Server. This creates a **/tmp/exports/** directory that contains all the exported data, ready to import and recreate within the Satellite 6 Server. As part of the import process, use the **`--directory`** option with the **`hammer import`** commands to specify this directory as the source directory.

IMPORTANT

Ensure you have sufficient disk space to extract the archive within the **/tmp/** directory.

The transition process uses local disk mirrors to import data from the archive into Satellite 6. If the **/tmp/** directory cannot be used, ensure that you use an alternative that provides sufficient space and read access for the **apache** user and group.

On Satellite 6, ensure that the **apache** user and group has read access to the **/tmp/exports/** directory. If necessary, adjust the group and permissions:

```
# chgrp -R apache /tmp/exports/
# chmod -R 0750 /tmp/exports/
```

IMPORTANT

If SELinux is enabled, ensure the **`tmp_t`** SELinux file context is applied to the **/tmp/exports/** directory. If necessary, apply the label manually:

```
# chcon -R system_u:object_r:tmp_t:s0 /tmp/exports/
```

If the SELinux context is not set correctly, Satellite cannot synchronize the content.

4.5. IMPORTING TO SATELLITE 6

Importing Satellite 5 data to a fresh Satellite 6 installation can be a lengthy process, requiring multiple steps, and the order in which you import the different entities is important. Some entities depend on the existence of others, and if you attempt to import entities in the wrong order, the import could fail. You also need to know exactly which exported CSV file is appropriate for which entity.

Options for Importing Entities

You can perform either a manual or an unattended import. Satellite 6 provides the **`hammer import all`** command to facilitate each of these approaches. This command is a wrapper for the various subcommands that import the different groups of entities. It calls each of the required subcommands in the correct order to complete the import of entities from Satellite 5 to a fresh Satellite 6 installation. You can perform any required setup or manipulation of your Satellite 5 data, create the required manifests, and then import everything in one step.

The **`hammer import all`** command ensures that entities with dependencies are created in the correct order. You can use the **`hammer import all --list-entities`** command to display all entities that the import process is aware of. This command also lists the entities in order of dependency. This is useful if you want to limit which entities to import (using the **`--entities`** parameter), and also if you choose to perform a manual import, and need to know the correct order in which to run the individual import commands (using the **`--dry-run`** parameter).

4.5.1. Prerequisites

The prerequisites for performing either a manual or an unattended import are the same:

- Valid output of **`satellite-export all`** from your Satellite 5 Server copied to the Satellite 6 Server and extracted into a suitable directory. The default location is **/tmp/exports**, but you can override this with the **`--directory`** parameter.
- New manifests for any organizations that you need to create in Satellite 6. There is no default manifest directory; specify the directory with the **`--manifest-directory`** parameter.

> **NOTE**
>
> If you do not use **`--manifest-directory`**, you need to import the manifests before you run the **`hammer import all`** command.

4.5.2. The Import Workflow

After the data exported from Satellite 5 is made available to the Satellite 6 system, it is ready to import. The import workflow is as follows:

1. Import organizations. This includes importing a manifest if one exists.
2. Import users.
3. Import system groups as host collections.
4. Enable and synchronize Red Hat repositories. This is referred to as repository discovery.
5. Import repositories.
6. Import custom channels and cloned channels as content views.
7. Import activation keys.
8. Import kickstart snippets as template snippets.
9. Import configuration files to Puppet modules.
10. Import system profiles as content hosts.

4.5.3. Performing an Unattended Import

This section describes how to use the **`hammer import all`** command to perform an unattended import of entities, previously extracted from Satellite 5 (for example, using the **`spacewalk-export`** command) as described in Section 4.4.2, "Transferring Exports to Satellite 6" .

On Satellite 6, use a command as follows to import the data previously extracted to the **/tmp/exports** directory:

```
# hammer import all --directory=/tmp/exports
```

If any errors occur during the import, inspect the **~/import.log** file for details. You can also use the **`--logfile`** ***`filename`*** parameter when you enter the **`import`** command to specify a different log file location.

4.5.3.1. Fine-tuning the Import Process

The **`hammer import all`** command passes information to each of the import subcommands. Unless otherwise specified, this command uses default values to perform the import. You can use the following parameters to help fine-tune the import:

- **`--into-org-id`** is passed to **`import organization --into-org-id`**.
- **`--manifest-directory`** is passed to **`import organization --upload-manifests-from`**.
- **`--merge-users`** is passed to **`import user --merge-users`**.
- **`--macro_mapping`** is passed to **`import config-file --macro-mapping`**.
- **`--debug`**, **`--delete`**, **`--directory`**, **`--logfile`**, **`--quiet`**, and **`--verbose`** work as for all other **`hammer import`** commands.
- **`--synchronize`** is passed to the **`--synchronize`** parameter for the following subcommands: **`repository-enable`**, **`repository`**, and **`content-view`**.
- **`--no-async`** is passed to the **`--no-async`** parameter for the following subcommands: **`repository-enable`**, **`repository`**, and **`content-view`**.
- **`--wait`** is passed to the **`--wait`** parameter for the following subcommands: **`repository-enable`**, **`repository`**, and **`content-view`**.

IMPORTANT

Red Hat recommends that you **always** use the **`--synchronize`**, **`--wait`**, and **`--no-async`** parameters.

If **`--synchronize`** is not specified, the **`import`** command does not synchronize content from Red Hat (**`repository-enable`**) or non-Red Hat (**`repository`**) sources. Content Views and Content Host creation depend on having content available, and will fail if synchronization is incomplete.

If **`--wait`** is not specified, the **`import all`** command does not wait until repositories have finished synchronizing. This will result in errors when the tool attempts to create content views and content hosts, because those entities rely on content being available.

If **`--no-async`** is not specified, the **`import all`** command makes all of its requests to Satellite 6 in parallel. This can overload the Satellite 6 Server and cause a variety of errors, such as connection timeouts and database-connection refusals.

The **`--synchronize`**, **`--wait`**, and **`--no-async`** parameters do not take arguments; if they are not specified on the command line, they default to false.

If an import-entity-command is not exposed by **`import all`**, the default for that parameter of that entity is used. For example, for **`import user`**, the role mapping at **/etc/hammer/cli.modules.d/import/role_map.yml** is always used.

4.5.4. Performing a Manual Import

This section describes how to perform a manual import of entities from Satellite 5 to a new installation of Satellite 6.

IMPORTANT

The order in which you import entities is important. Entities are owned by organizations in Satellite 5; consequently, you need to import organizations before users, for example. You can use the **`hammer import all --dry-run`** command to list the available entities and the order in which they should be imported.

4.6. IMPORTING ORGANIZATIONS

To transition organizations, use the **users.csv** file and recreate the Satellite 5 organizations listed within it. You can use the **`hammer import`** command on the command line or use the hammer interactive shell.

Run the following command to import organizations into Satellite 6:

```
# hammer import organization --csv-file /tmp/exports/users.csv
```

The following is an example of an interactive session. This example also demonstrates the **`--upload-manifests-from`** and **`--verbose`** options:

Example 4.2. Example of Interactive Import Session

```
# hammer shell
```

```
hammer> import organization --csv-file /tmp/exports/users.csv \
--upload-manifests-from /root/manifests --verbose

Importing from /tmp/exports/users.csv
Creating new organization: RED HAT SATELLITE ENGINEERING
Uploading manifest /root/manifests/RED_HAT_SATELLITE_ENGINEERING.zip to
org-id 5
Waiting for the task [a231d19c-aee7-42b8-9566-07651ac029f4] ......
Organization [1->5] already imported.
Organization [1->5] already imported.
Organization [1->5] already imported.
Organization [1->5] already imported.
Creating new organization: SOE-ORG
Uploading manifest /root/manifests/SOE-ORG.zip to org-id 6
Waiting for the task [5da6dd16-0bf6-4ad0-924f-a9d5e1802565] ......
Organization [7->6] already imported.
Summary
  Found 5 organizations.
  Created 2 organizations.
  Uploaded 2 manifests.
```

Use the **`hammer organization list`** command to list the organizations within Satellite 6.

```
# hammer organization list
---|-------------------------------|-------------------------------|--
----------
ID | NAME                          | LABEL                         |
DESCRIPTION
---|-------------------------------|-------------------------------|--
----------
5  | RED HAT SATELLITE ENGINEERING | RED_HAT_SATELLITE_ENGINEERING |
6  | Test Organization             | Test_Organization             |
3  | Satellite Documentation       | Satellite_Documentation       |
---|-------------------------------|-------------------------------|--
----------
```

The import process creates organizations based on the organizations listed in the **user.csv** file. The Satellite 5 organization IDs are mapped to new Satellite 6 organization IDs. This is illustrated by the " [1→5]" and similar entries in Example 4.2, “Example of Interactive Import Session” . Alternatively, you can use the **`hammer import organization --into-org-id`** ***`org_id`*** command to reduce all of the Satellite 5 organizations into a single, flat organization within Satellite 6. You can use the **`hammer organization list`** command to determine the correct organization ID.

WARNING

All import data is stored in CSV files in the **~/.transition_data/** directory. This information is critical for any subsequent data imports. Do not modify the data in this directory.

A history of hammer commands is stored in the **/root/.foreman/history** file, and any errors from hammer commands are stored in the **/root/.foreman/log/hammer.log** file.

The **`hammer import`** command logs all output to the **~/import.log** file. You can use the **`--logfile`** option to any **`hammer import`** subcommand to specify a different name and location for the log file.

Generating and Activating a Manifest

You need to activate a manifest for the organizations for which you want to populate content and other data. You can generate a manifest from within the Red Hat Customer Portal, and then activate that manifest for the imported organization.

The following procedure assumes you have already created a suitable manifest in the Red Hat Customer Portal.

To Activate the Manifest for Satellite 6:

1. Log in to the Satellite 6 web UI as an administrative user.
2. Select the required organization from the main menu at the upper left.
3. Click **Content** > **Red Hat Subscriptions**
4. On the **Actions** tab, under **Upload New Manifest**, click **Browse**, navigate to and select the manifest file that you downloaded.
5. In the Satellite 6 web UI, click **Upload** to upload the manifest to the Satellite 6 Server.

Repeat this procedure for each required organization.

4.6.1. Importing Users

The import process recreates the Satellite 5 users in each Satellite 5 organization from the **users.csv** file. User passwords cannot be copied from Satellite 5 to Satellite 6; instead, the process generates a new random password for each user that it imports. This information is saved in a CSV file, which as the administrator you can parse and send notifications to each user listed with their new password. You can use the **`hammer import`** command on the command line or use the hammer interactive shell.

The import process tracks which entities have already been processed. You can, for example, enter the **`hammer import user`** command multiple times, using different input files (CSV files). The **`import`** command recognizes user IDs it has already imported, and skips them on subsequent processes.

Run the following command to import users into Satellite 6:

```
# hammer import user --csv-file /tmp/exports/users.csv \
--new-passwords passwords.csv
```

The following is an example of an interactive session:

Example 4.3. Example of Interactive Import Session

```
hammer> import user --csv-file /tmp/exports/users.csv --new-passwords
/root/new-user-passwords.csv
Summary
  Created 7 users.
```

To show the users were created, log in to the web UI and navigate to **Administer** > **Users**. Alternatively, use the command line as shown below. You can list all users or search by name or other properties.

```
# hammer user list

---|-------------------|-------------|-------------------
ID | LOGIN             | NAME        | EMAIL
---|-------------------|-------------|-------------------
4  | sat5_admin        | Admin Nimda | root@localhost
3  | admin             | Admin User  | root@milky.way
6  | normal            | Morm Al     | root@localhost
10 | testorg           | Red Hat     | root@localhost
7  | sysgroup          | Red Hat     | root@localhost
8  | suseadmin         | SUSE Admin  | root@localhost
---|-------------------|-------------|-------------------


# hammer user list --search sat5_admin

---|------------|-------------|---------------
ID | LOGIN      | NAME        | EMAIL
---|------------|-------------|---------------
4  | sat5_admin | Admin Nimda | root@localhost
---|------------|-------------|---------------
```

The following is an example password file for the imported users:

```
# head new-user-passwords.csv

mail,login,password

sat5admin@example.com,sat5_admin,sat5_admin_svenkmxf
auser1@example.com,auser1,auser1_pwfnagdk
auser2@example.com,auser2,auser2_rsgywazf
```

NOTE

By default, Satellite 6 creates an **admin** user as the initial administrator. It is common for Satellite 5 customers to also create a generic administrative user; consequently, if the import process detects a Satellite 5 **admin** user, it renames that user to **sat5_admin**.

4.6.2. Transitioning System Groups to Host Collections

Red Hat Satellite 5 uses system groups to maintain collections of multiple systems. Satellite 6 uses host collections to perform the same task. The Satellite transition tools provide options to transition your Satellite 5 system groups to Satellite 6 host collections.

The following example demonstrates a typical use case. Run this command on your Satellite 6 system, and substitute the CSV file with your own:

```
# hammer import host-collection --csv-file /tmp/exports/system-groups.csv
Summary
  Created 4 host_collections.
```

You can use the **`hammer host-collection list --organization-id`** ***`ORG-ID`*** to verify that the system groups have been recreated as host collections. Ensure you use the correct organization ID.

You can also log in to the web UI as an administrator to verify that the host collections were created. Ensure you are in the correct organization, and then navigate to **Hosts** > **Host Collections**.

4.6.3. Enabling Yum Repositories

Red Hat Satellite 5 and Satellite 6 use different methods to group RPM files and to present them. In Satellite 5, all RPM files are placed into channels which you clone, and you then manage the content from within those cloned channels. The **`satellite-sync`** command synchronizes Red Hat channels from RHN into Satellite 5. In Satellite 6, everything is a yum repository and you filter the content from that repository that is exposed to the systems being managed by Satellite 6.

Several options exist for recreating Satellite 5 repositories and importing the content:

- Use the **`hammer import repository --synchronize`** command to initiate repository synchronization in the background.
- Use the **`hammer import repository --synchronize --wait`** command to initiate repository synchronization in the background, but wait for each synchronization to complete before proceeding to the next repository.
- Use the **`hammer import repository`** command with neither option and then synchronize the content manually using the web UI or **`hammer`** commands.

Recommended Practices for Synchronizing Content

The default behavior of the **`hammer import repository`** command is to **not** automatically synchronize content. Many of the following import operations require that synchronization be complete before they can run successfully. Synchronization is a time-consuming task, because it involves retrieving content from external sources into your Satellite 6 Server.

Red Hat recommends that you schedule a background synchronization of all content before proceeding. Only repositories available to imported organizations are enabled. The **`hammer import`** command does not alter organizations that are not part of the overall transition process.

Run the following command to find, enable, and synchronize all Red Hat content enabled by the uploaded manifest files and matched by channels synchronized on the Satellite 5 Server. This command tells the import process to wait until the synchronization is complete, and only synchronizes one repository at a time.

```
# hammer import repository-enable --synchronize --wait --no-async
```

After the **`repository-enable`** command is complete, enter the following command to process local and custom repositories in a similar fashion.

```
# hammer import repository --synchronize --wait --no-async
```

The following is an example session of importing repositories but without performing any content synchronization.

Example 4.4. Example Session of Importing Repositories

The following is an example session of successfully importing repositories into Satellite 6.

```
# hammer import all --entities repository
Import repository           with arguments --csv-file
/tmp/exports/repositories.csv
[Foreman] Username: admin
[Foreman] Password for admin:
Summary
Created 10 repositories.
Created 5 products.
```

Enabling External Yum Repositories

You need to recreate all non-Red Hat yum repositories that Satellite 5 was configured to use as new external yum repositories from which Satellite 6 can import content.

The following example demonstrates using the **`--synchronize`** option to import and synchronize repositories.

NOTE

A successful import and synchronization produces very little output unless you use the **`-v`** (verbose) option. The synchronization process can take a long time (several hours depending on the number of repositories, bandwidth, and other factors), during which no output is displayed.

Example 4.5. Importing and Synchronizing non-Red Hat Repositories

```
# hammer import repository --synchronize --wait \
--no-async --csv-file /tmp/exports/repositories.csv

Summary
  No action taken.
```

Viewing Imported Products and Repositories

Navigate to **Content** > **Products** in the Satellite 6 web UI to view the resulting Products and repositories for your organization. Alternatively, use the following **`hammer`** commands:

```
# hammer organization list

---|-------------------------------|-------------------------------|--
----------
ID | NAME                          | LABEL                         |
DESCRIPTION
---|-------------------------------|-------------------------------|--
----------
5  | RED HAT SATELLITE ENGINEERING | RED_HAT_SATELLITE_ENGINEERING |
6  | Test Organization             | Test_Organization             |
3  | Satellite Documentation        | Satellite_Documentation        |
---|--------------------------------|--------------------------------|--
----------

# hammer product list --organization-id 5

----|--------------|-------------|-------------------------------|-----
---------|-----------------
ID  | NAME         | DESCRIPTION | ORGANIZATION                  |
REPOSITORIES | SYNC STATE
----|--------------|-------------|-------------------------------|-----
---------|-----------------
131 | GOOGLE.COM   |             | RED HAT SATELLITE ENGINEERING | 1
| Syncing Complete.
133 | NOVELL.COM   |             | RED HAT SATELLITE ENGINEERING | 3
| Syncing Complete.
134 | OPENSUSE.ORG |             | RED HAT SATELLITE ENGINEERING | 2
| Syncing Complete.
135 | PNL.GOV      |             | RED HAT SATELLITE ENGINEERING | 1
| Syncing Complete.
132 | REDHAT.COM   |             | RED HAT SATELLITE ENGINEERING | 3
| Syncing Complete.
----|--------------|-------------|-------------------------------|-----
---------|-----------------
```

IMPORTANT

The default behavior of the **`hammer import repository`** command is to not automatically synchronize content, but rather only to enable the repositories listed in the *repositories.csv* file. Red Hat recommends that you use the **`--synchronize`** option to schedule a background synchronization of all content before proceeding.

Enabling Red Hat Repositories

The Red Hat repositories are synchronized from the Red Hat Content Delivery Network (CDN). Because of these differences, it can be difficult to determine if you have imported all relevant Red Hat content into your Satellite 6 Server. Red Hat recommends that you import into Satellite 6 all CDN content that equates to previously synchronized Satellite 5 channels. This ensures that your Red Hat Enterprise Linux systems have access to the same content as they did in Satellite 5.

The **`hammer import`** command on Satellite 6 provides an option to review your Satellite 5 channels and enable the corresponding repositories in Satellite 6. This section guides you through the process of discovering and enabling the appropriate Red Hat repositories. This command provides the same **`--wait`** and **`--synchronize`** options for initiating the import of Red Hat content automatically. This needs to be completed before moving to the next step.

The following example demonstrates the **`--synchronize`** option to enable and synchronize Red Hat repositories.

Example 4.6. Enabling and Synchronizing Red Hat Repositories

```
# hammer import repository-enable --csv-file /tmp/exports/channels.csv \
--synchronize

Only repositories available to IMPORTED organizations will be enabled!
```

```
Organization ACME_Corporation...
Organization GLOBAL_SUPPORT_SERVI_RED_HAT__INC_...
Enabling /content/dist/rhel/server/5/5Server/x86_64/os/Packages for
channel rhel-x86_64-server-5
Sync started!

Enabling
/content/dist/rhel/server/5/5Server/x86_64/supplementary/os/Packages for
channel rhel-x86_64-server-supplementary-5
Sync started!

Enabling /content/dist/rhel/server/6/6Server/x86_64/rhn-
tools/os/Packages for channel rhn-tools-rhel-x86_64-server-6
Sync started!

Enabling /content/dist/rhel/server/6/6Server/x86_64/optional/os/Packages
for channel rhel-x86_64-server-optional-6
Sync started!

Enabling /content/dist/rhel/server/6/6Server/x86_64/os/Packages for
channel rhel-x86_64-server-6
Sync started!

Enabling /content/dist/rhel/server/5/5Server/x86_64/rhn-
tools/os/Packages for channel rhn-tools-rhel-x86_64-server-5
Sync started!
```

4.6.4. Transitioning Custom and Cloned Channels to Content Views

Red Hat Satellite 5 uses the concept of cloned channels to restrict the number of Red Hat-supplied channels and RPM files. Custom content is added to custom channels, which are typically children of cloned channels. In Satellite 6, for both Red Hat and non-Red Hat content, you use a single repository and then provide filtered views of that repository to the systems managed by Satellite 6. These filtered views are called content views.

This section describes how to recreate your Satellite 5 custom and cloned channels as content views. This process provides a near-equivalent set of content to systems in Satellite 6 as was available in Satellite 5.

Run the following command to import the Satellite 5 channels as content views in the Satellite 6 Server. Use the **`--dir`** option to specify the appropriate export directory.

```
# hammer import content-view --csv-file /tmp/exports/CHANNELS/export.csv \
--dir /tmp/exports/CHANNELS
```

Run the following command to synchronize the imported channels:

```
# hammer import content-view --csv-file /tmp/exports/CHANNELS/export.csv \
--synchronize
```

You can combine the two commands and synchronize the channels at the same time.

```
# hammer import content-view --csv-file /tmp/exports/CHANNELS/export.csv \
--dir /tmp/exports/CHANNELS --synchronize
```

The following is an example session of importing channels to content views.

Example 4.7. Importing Channels to Content Views

```
# hammer import content-view --csv-file /tmp/exports/CHANNELS/export.csv
\
--dir /tmp/exports/CHANNELS

Creating new product: Local-repositories
Creating new local_repository: Local repository for clone-rhel-x86_64-
server-5
No such content_view: 101
Repository Local_repository_for_clone-rhel-x86_64-server-5 is not
(fully) synchronized. Retry once synchronization has completed.

Product [1Local-repositories->118] already imported.
Creating new local_repository: Local repository for clone-rhel-x86_64-
server-6
No such content_view: 102

Product [1Local-repositories->118] already imported.
Creating new local_repository: Local repository for custom-clone-master-
puppet-rhel6-server-x86_64
Repository Local_repository_for_custom-clone-master-puppet-rhel6-server-
x86_64 is not (fully) synchronized. Retry once synchronization has
completed.

Product [1Local-repositories->118] already imported.
Creating new local_repository: Local repository for epel-puppet-rhel6-
server-x86_64
Repository Local_repository_for_epel-puppet-rhel6-server-x86_64 is not
(fully) synchronized. Retry once synchronization has completed.
```

Example 4.8. Synchronizing Imported Channels

```
# hammer import content-view --csv-file /tmp/exports/CHANNELS/export.csv
\
--synchronize

Product [1Local-repositories->118] already imported.
Local_repository [1117->12] already imported.
Sync started!
No such content_view: 101
Repository Local_repository_for_clone-rhel-x86_64-server-5 is not
(fully) synchronized. Retry once synchronization has completed.

Product [1Local-repositories->118] already imported.
Local_repository [1113->13] already imported.
Sync started!
No such content_view: 102
```

```
Repository Local_repository_for_clone-rhel-x86_64-server-6 is not
(fully) synchronized. Retry once synchronization has completed.

Product [1Local-repositories->118] already imported.
Local_repository [1115->14] already imported.
Sync started!
Repository Local_repository_for_custom-clone-master-puppet-rhel6-server-
x86_64 is not (fully) synchronized. Retry once synchronization has
completed.

Product [1Local-repositories->118] already imported.
Local_repository [1125->21] already imported.
Sync started!
```

Example 4.9. Combining Import and Synchronization of Channels

```
# hammer import content-view --csv-file /tmp/exports/CHANNELS/export.csv
\
--dir /tmp/exports/CHANNELS --synchronize

Product [1Local-repositories->98] already imported.
Local_repository [1117->12] already imported.
No such content_view: 101
Creating new content_view: Clone of Red Hat Enterprise Linux (v. 5 for
64-bit x86_64)

Product [1Local-repositories->98] already imported.
Local_repository [1113->13] already imported.
No such content_view: 102
Creating new content_view: Clone of Red Hat Enterprise Linux Server (v.
6 for 64-bit x86_64)

Product [1Local-repositories->98] already imported.
Local_repository [1115->14] already imported.
No such content_view: 103
Creating new content_view: Clone of RHEL Server Optional (v. 6 64-bit
x86_64)
```

You can use the web UI to monitor the status of the import. Log in as an administrator, select the appropriate organization from the context menu at the upper left, and then click **Content** > **Sync Status**.

You might need to enter the command with the `synchronize` option more than once after the previous steps have completed, before you see the content view being generated.

4.6.5. Importing Activation Keys

Activation keys in both Satellite 5 and 6 serve almost identical purposes. The command-line tools use activation keys to register and subscribe systems to Red Hat Satellite. Activation keys in Satellite 5 have different properties, such as system groups, channel entitlements, and system entitlements (provisioning, monitoring), compared to Satellite 6, which uses content views and host collections.

The earlier stages of the transition process focused on recreating the Satellite 5 data types within Satellite 6. This section describes how to import the activation keys from Satellite 5 and to associate them with the imported data types.

The following example illustrates a typical use case for importing activation keys.

Example 4.10. Importing Activation Keys into Satellite 6

```
# hammer import activation-key --csv-file /tmp/exports/activation-
keys.csv

Activation key usage_limit: unlimited
Creating new activation_key: 1-rhel5-puppet
Associating activation key [1] with host collections [2]
Creating new ak_content_view: ak_1
Publishing content view: 15
Associating activation key [1] with content view [15]
Updating activation_key with id: 1
Creating new ak_content_view: ak_2
.
.
.
Summary
  Skipped 2 activation_keys.
  Created 3 activation_keys.
```

You can use the **`hammer activation-key list --organization-id`** ***`ORG-ID`*** command to verify that the activation keys have been recreated. Ensure you use the correct organization ID.

You can also log in to the web UI as an administrator to verify that the activation keys were created. Ensure you are in the correct organization, and then navigate to **Content** > **Activation Keys**.

The following example illustrates the use of the command-line tools to verify the import of activation keys.

Example 4.11. Verifying the Import of Activation Keys

```
# hammer activation-key list --organization-id 3

---|------------------------------------|----------------|----------
-------------|-------------
ID | NAME                               | CONSUMED       | LIFECYCLE
ENVIRONMENT | CONTENT VIEW
---|------------------------------------|----------------|----------
-------------|-------------
1  | 1-rhel6-puppet                     | 0 of Unlimited |
|ak_1
3  | 1-rhel5-puppet                     | 0 of Unlimited |
|ak_3
---|------------------------------------|----------------|----------
-------------|-------------
```

4.6.6. Transitioning Kickstart Profiles

The Satellite transition tools do not migrate entire kickstart profiles. Due to some significant differences between the Satellite 5 and Satellite 6 infrastructures, there is no suitable migration path between the two versions.

The transition tools can migrate kickstart **scripts**, and these can be used in Satellite 6 Provisioning Templates, which are an approximation of kickstart profiles.

Exporting Kickstart Scripts

The transition tools extract and export kickstart scripts from kickstart profiles as part of the overall export process on the Satellite 5 Server. These scripts are stored in the **kickstart-scripts.csv** file in the export archive, which is copied to the Satellite 6 Server.

Importing Template Snippets

When you extract the export archive onto the Satellite 6 Server, the **`hammer import`** command uses the **kickstart-scripts.csv** file to create template snippets. These snippets are in the form of %pre and %post scripts that you can use with any new kickstart profiles in Satellite 6.

4.6.7. Transitioning Configuration Channels to Puppet Modules

Red Hat Satellite 5 uses configuration channels to support configuration file management. Configuration channels are collections of configuration entries, which in turn specify files and values that Satellite 5 applies to systems that are registered to it. Configuration channels support the upload of flat files for delivery to associated systems.

Satellite 6 uses Puppet to provide improved configuration management; consequently, part of the transition process requires that configuration channels be converted into a single Puppet module.

The export process on Satellite 5 includes the latest revision of all configuration files. The import process uses this information to perform the following tasks:

- Generate Puppet modules for each Satellite 5 configuration channel.
- Map any Satellite 5 macros in the configuration files to whatever Puppet facts are found.
- Build the modules.
- Create a Satellite 6 repository for each configuration channel, and a Product to hold those repositories.
- Upload the built Puppet modules into Satellite 6.

The following table describes the current mapping between Satellite 5 substitution macros and Puppet facts.

Table 4.1. Mapping of Satellite 5 Macros to Satellite 6 Puppet Facts

Satellite 5 Macro	Puppet Fact
rhn.system.sid	None
rhn.system.profile_name	None

Satellite 5 Macro	Puppet Fact
rhn.system.description	None
rhn.system.hostname	FQDN or host name
rhn.system.ip_address	ipaddress
rhn.system.custom_info(key_name)	None
rhn.system.net_interface.ip_address(eth_device)	ipaddress_{NETWORK INTERFACE}
rhn.system.net_interface.netmask(eth_device)	netmask_{NETWORK INTERFACE}
rhn.system.net_interface.broadcast(eth_device)	None
rhn.system.net_interface.hardware_address(eth_device)	macaddress_{NETWORK INTERFACE}
rhn.system.net_interface.driver_module(eth_device)	None

Transitioning Configuration Channels to Puppet Modules

The following example illustrates a simple use case for transitioning configuration channels to Puppet modules.

Example 4.12. Transitioning Configuration Channels and Files to Puppet Modules

```
# hammer import config-file --csv-file /tmp/exports/config-files-latest.csv

Writing converted files
Building and uploading puppet modules
Successfully uploaded file 'redhatsatelliteengineering-test_channel-1.0.0.tar.gz'.
Summary
  Uploaded 1 puppet module.
  Wrote 1 puppet module.
  Wrote 4 puppet_files.
  Created 1 puppet repository.
  Created 1 product.
```

4.6.8. Transitioning System Profiles to Content Hosts

Red Hat Satellite 5 uses System Profiles to store information about the systems (machines) that are registered to it. This includes the operating system version and a list of all packages that are installed on each system. This applies to both bare-metal and virtual machines.

In Satellite 6, this information is stored in Content Hosts. Instead of "system", Satellite 6 uses the term "host" to refer to any machine that is registered to it. Part of the transition process maps Satellite 5 System Profiles to Satellite 6 Content Hosts. A Satellite 6 host is created when an actual system re-registers to Satellite 6 with the consumer ID of the Content Host.

Transitioning system profiles is the last step in the transition process prior to actually transitioning your Satellite 5 systems to Satellite 6. You can transition system profiles as part of the **`hammer import all`** command, or by specifically importing them, using the **`hammer import all --entities content-host`** command.

Building the System Profile Transition RPM File

Before you can transition your Satellite 5 systems to Satellite 6, you need to build the required RPM file. The **`hammer import all`** command displays instructions on how to do this, tailored to your specific environment, after you have successfully transitioned system profiles to content hosts.

After you have successfully transitioned your system profiles to content hosts, you should see output similar to the following:

Example 4.13. Example Instructions for Building a System Profile Transition RPM File

To build the system-profile-transition rpm, enter:

```
# cd /root/rpm-working-dir/SPECS && rpmbuild -ba \
--define "_topdir /root/rpm-working-dir" \
system-profile-transition-myhost.example.com-1410140956-0.0.1.spec
```

Then find your system-profile-transition-myhost.example.com-1410140956 package in /root/rpm-working-dir/RPMS/noarch/ directory.

4.6.9. Transitioning Systems

This section covers background information, prerequisites, and assumptions that must be addressed prior to transitioning your client systems. It also covers the actual process for transitioning your Satellite 5 client systems to Satellite 6 client hosts. This process is primarily aimed at customers who have numerous systems currently registered to a Satellite 5 Server.

The system transition process aims to achieve the following goals:

- Ensure that each Satellite 5 client is registered to a new Satellite 6 instance, with as much of the existing Satellite 5 setup maintained as possible.
- Help customers stay in compliance with their Red Hat Enterprise Linux usage. The process removes the Satellite 5 entitlements used by a given system when that system is registered to Satellite 6.

Prerequisites

Before you begin the system transition process, ensure that you have each of the following:

- An up-to-date Satellite 5.6 or 5.7 Server. In this example, this system is called "SAT5x".
- An up-to-date Red Hat Enterprise Linux client system registered to SAT5x.
- An up-to-date Satellite 6 Server. In this example, this system is called "SAT6".

- Manifests created for each Organization on SAT5x and downloaded to the Satellite 6 base system. Manifests renamed according to the organization for which it is intended. Punctuation marks and spaces replaced with an underscore. For example, an organization called MyOrg, Inc. can use the **MYORG__INC_.zip** manifest.

Assumptions

The system migration process makes several assumptions as part of the task of moving your Satellite 5 systems to Satellite 6:

- The **`hammer import content-host`** command has completed successfully.
- An RPM file containing a mapping between Satellite 5 SIDs and matching Satellite 6 content-host UUIDs has been created.
- The above-mentioned RPM file has been installed on the Satellite 5 Server's base system.
- The **`sat5to6`** package and dependencies have been installed on each client system. This package provides the **`sat5to6`** command, which performs the following tasks on the system where it is used:
- Queries its Satellite 5 parent for a consumer ID.
- Loads the appropriate PEM files onto the client machine, based on the Satellite 5 channels to which the machine is subscribed.
- Registers the host on which it is used to a specified Satellite 6 Server, attaching it to the content host UUID specified by its Satellite 5 parent.
- Manages the "retired" Satellite 5 profile in one of three ways, as specified by the user:
- **`keep`**: Do nothing. Retain the Satellite 5 profile in its original state.
- **`unentitle`**: Retain the Satellite 5 profile but remove all subscriptions and entitlements (default).
- **`purge`**: Delete the Satellite 5 profile completely.

Performing the Transition Using the sat5to6 Script

This section describes the process for transitioning your Satellite 5 client systems to Satellite 6 client hosts, and updating the subscription configuration accordingly based on the profiles.

This section makes the following assumptions:

- You have successfully used the **`spacewalk-export`** command on your Satellite 5 Server to export all required entities.
- You have copied the resulting **spacewalk_export.tar.gz** file to the **/tmp/** directory on your Satellite 6 Server, and extracted it into the same directory.
- You have successfully used the **`hammer import all`** command on the Satellite 6 Server, and followed the resulting instructions to build the required RPM file.
- You have the necessary system transition packages installed on the Satellite 5 Server. To ensure these packages are installed, enter the following command:

  ```
  # yum install /tmp/system-profile-transition-*.rpm
  ```

■

Transitioning Your Satellite 5 Systems to Satellite 6

This process assumes that your client is already subscribed to the appropriate RHN Tools channel. This channel provides access to the **sat5to6** RPM file and its dependencies. For each client that is registered to your Satellite 5 Server, enter the following commands:

1. Install the **sat5to6** package and its dependencies:

   ```
   # yum install sat5to6
   ```

2. Install the Public Certificate for Satellite's Certificate Authority. This also configures **subscription-manager** with the correct URL so that the system can properly register via the Satellite 6.x instance. In this example, $FQDN represents the fully qualified domain name of the Satellite or Satellite Capsule.

   IMPORTANT

   Ensure you use HTTP and not HTTPS for this installation; the Satellite 6 CA certificate is not yet installed and so HTTPS will fail.

   ```
   # yum install http://$FQDN/pub/katello-ca-consumer-latest.noarch.rpm
   ```

3. Update the **yum** and **openssl** packages.

   ```
   # yum update yum openssl
   ```

4. Use the **sat5to6** command to register your client to your Satellite 6 instance, and attach it to the content host created for it by the import process.

   ```
   # sat5to6 --registration-state unentitle \
   --legacy-user=admin --legacy-password=password \
   --destination-user=admin --destination-password=changeme
   ```

5. Enable the Satellite Tools repository which contains the **katello-agent** and **puppet** packages.

   ```
   # subscription-manager repos --enable=rhel-*-server-satellite-tools-
   6.2-rpms
   ```

6. Install the **katello-agent** and **puppet** packages:

   ```
   # yum install katello-agent puppet
   ```

7. Configure the **goferd** and **puppet** services to start on boot.

   ```
   # chkconfig goferd on
   # chkconfig puppet on
   ```

8. Configure Puppet with the host name of the Satellite or Satellite Capsule that will manage its configuration. In this example, $FQDN represents the fully qualified domain name of the Satellite or Satellite Capsule.

```
# echo "server=$FQDN" >> /etc/puppet/puppet.conf
```

9. Restart the **goferd** and **puppet** services.

```
# service goferd restart
# service puppet restart
```

NOTE

As part of the new Satellite 6 installation, the **`katello-ca-consumer-latest`** package sets up the CA certificates and default values in the **/etc/rhsm/rhsm.conf** file. This enables SSL and also means you do not need to specify the **`--destination-url`** parameter as part of the **`sat5to6`** command.

Example 4.14. Example Session of a System Transition

The following is an example session of a successful transition of Satellite 5 systems to Satellite 6 hosts. The actual output will vary according to the organization names, system names, and other details of your own environment.

```
# sat5to6 --registration-state=unentitle --legacy-user=admin \
--legacy-password=password --destination-user=admin \
--destination-password=changeme

Org: MY ENGINEERING ORG
Environment: Library

Retrieving existing legacy subscription information...

+-----------------------------------------------------+
System is currently subscribed to these legacy channels:
+-----------------------------------------------------+
rhel-x86_64-server-6

+-----------------------------------------------------+
Installing product certificates for these legacy channels:
+-----------------------------------------------------+
rhel-x86_64-server-6

Product certificates installed successfully to /etc/pki/product.

Attempting to register system to destination server...
WARNING

This system has already been registered with Red Hat using RHN Classic.

Your system is being registered again using Red Hat Subscription
Management. Red Hat recommends that customers only register once.

To learn how to unregister from either service please consult this
Knowledge Base Article: https://access.redhat.com/kb/docs/DOC-45563
The system has been registered with ID: ac063466-0abc-1bb3-abc0-
33abf6c1dd3a
```

```
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux Server
Status:       Subscribed

System 'dhcp1234.example.com' successfully registered.
```

CHAPTER 5. TRANSITIONING TO THE SATELLITE 6 API

One of the many differences between Red Hat Satellite 5 and Satellite 6 is the API. Satellite 5 uses an XMLRPC-based API. Satellite 6 uses a REST-based API. This fundamental difference requires that any existing scripts or tools that have been integrated with the Satellite 5 API must be reviewed and at least partially rewritten before use with the Satellite 6 REST API.

This section provides some comparisons of how to achieve the same use case within each Product. This is not designed to be a tutorial in any specific programming language. Neither are the scripts secured over HTTPS. They provide a starting point for anyone maintaining Satellite 5 API scripts to start the transition to the Satellite 6 API.

IMPORTANT

The Satellite 6 transition tools do not transition the Satellite 5 API or scripts to Satellite 6. Use this section as a starting point to begin your own transition process.

Further Information

You can find further API documentation at the following locations on your own Satellite Servers:

- Satellite 6: https://satellite6.example.com/apidoc/v2.html
- Satellite 5: https://satellite5.example.com/rpc/api

5.1. EXAMPLE API SCRIPTS

The examples in this section cover the following:

1. Systems and hosts in Red Hat Satellite
   a. Authenticate and request a list of systems (Satellite 5) or hosts (Satellite 6) available to the user who logged in.
2. Users and Roles
   a. Authenticate and request a list of all users visible to the user who logged in.
   b. Delete the **example** user if it exists.
   c. Create a new **example** user and ensure that its role is set to *Administrator*.

This section provides a total of five different ways to achieve the same result; two examples for Satellite 5 and three examples for Satellite 6.

- A Satellite 5 Python script
- A Satellite 6 Python script
- A Satellite 6 Ruby script
- A Satellite 5 **spacecmd** example
- A Satellite 6 **hammer** example

NOTE

The **spacecmd** command is not a supported feature in Satellite 5.6. It is supported in Satellite 5.7, and is shown here for the sake of completeness.

A total of 10 examples are provided. The first examples cover the simpler use cases of listing systems and hosts, followed by the more complex examples covering users and roles.

5.1.1. Listing Systems and Hosts

The examples in this section describe different ways to list systems and hosts available to the user who logged in.

Using Python to List Available Systems on Satellite 5

This example uses a Python script to connect to Satellite 5, authenticate, and retrieve a list of available systems.

```
#!/usr/bin/python
import xmlrpclib

# Define Satellite location and login details
SATELLITE_URL = "http://localhost/rpc/api"
SATELLITE_LOGIN = "admin"
SATELLITE_PASSWORD = "password"

client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

# Authenticate and get session key
key = client.auth.login(SATELLITE_LOGIN, SATELLITE_PASSWORD)

# Get list of systems available to user
list = client.system.listSystems(key)
for system in list:
   print system.get('id')
   print system.get('name')

# Logout
client.auth.logout(key)
```

Using Python to List Available Hosts on Satellite 6

This example uses a Python script to connect to Satellite 6, authenticate, and retrieve a list of available hosts.

```
#!/usr/bin/python
import json
import requests

# Define Satellite location and login details
SAT_API = "https://localhost/api/v2/"
USERNAME = "admin"
PASSWORD = "changeme"
SSL_VERIFY = False
```

```
def get_json(location):
    """
    Performs a GET using the passed URL location
    """

    r = requests.get(location, auth=(USERNAME, PASSWORD),
verify=SSL_VERIFY)

    return r.json()

def main():
    # List all hosts available to the user
    hosts = get_json(SAT_API + "hosts/")

    # Pretty Print the returned JSON of Hosts
    print json.dumps(hosts, sort_keys=True, indent=4)

if __name__ == "__main__":

main()
```

Using Ruby to List Available Hosts on Satellite 6

This example uses a Ruby script to connect to Satellite 6, authenticate, and retrieve a list of available hosts.

```
#!/usr/bin/ruby193-ruby
require 'json'
require 'rest-client'

url = 'https://localhost/api/v2/'
$username = 'admin'
$password = 'changeme'

def get_json(location)
    response = RestClient::Request.new(
        :method => :get,
        :url => location,
        :user => $username,
        :password => $password,
        :headers => { :accept => :json,
        :content_type => :json }
    ).execute
    results = JSON.parse(response.to_str)
end

hosts = get_json(url+"hosts/")
#puts JSON.pretty_generate(hosts)

puts "Hosts within Satellite are:"
hosts['results'].each do |name|
      puts name['name']
end

exit()
```

Using the Command Line to List Available Systems on Satellite 5

Use the following command to list available systems on Satellite 5:

```
# spacecmd -u username -p password system_list
```

An example session might appear as follows:

```
# spacecmd -u admin -p password system_list

INFO: Spacewalk Username: admin
INFO: Connected to https://localhost/rpc/api as admin
test_02.example.com
```

Using the Command Line to List Available Hosts on Satellite 6

Use the following command to list available hosts on Satellite 6:

```
# hammer host list
```

An example session might appear as follows:

```
# hammer host list

[Foreman] password for admin:
---|------------------|--------------------|------------|--------------|-
-----------------
ID | NAME             | OPERATING SYSTEM   | HOST GROUP | IP           |
MAC
---|------------------|--------------------|------------|--------------|-
-----------------
1  | test.example.com | RHEL Server 6.5    |            | 10.34.34.235 |
e4:1f:13:6b:ed:0c
```

5.1.2. Deleting and Creating Users

The examples in this section describe different ways to locate, create, and delete users.

Using Python to Manage Users on Satellite 5

This example uses a Python script to connect to and authenticate against a Satellite 5 Server. It then goes on to search for a specific user (**example**), to delete that user if it exists, and then recreate it with Administrator privileges.

```
#!/usr/bin/python
import xmlrpclib

# Define Satellite location and login details
SATELLITE_URL = "http://localhost/rpc/api"
SATELLITE_LOGIN = "admin"
SATELLITE_PASSWORD = "password"

client = xmlrpclib.Server(SATELLITE_URL, verbose=0)
```

```
# Authenticate and get session key
key = client.auth.login(SATELLITE_LOGIN, SATELLITE_PASSWORD)

# Get list of users
list = client.user.list_users(key)
print "Existing users in Satellite:"
for user in list:
   print user.get('login')

# Look for user example and if found, delete the user
for user in list:
   if user.get('login') == 'example':
       deleteuser = client.user.delete(key, 'example')
       if deleteuser == 1:
           print "User example deleted"

# Create a user called example
createuser = client.user.create(key, 'example', 'password', 'Example',
'User', "root@localhost")
if createuser == 1:
    print "User example created"
    # Admin Org Admin role to the example user
    adminrole = client.user.addRole(key, 'example', 'org_admin')
    if adminrole == 1:
        print "Made example an Org Admin"

# Logout
client.auth.logout(key)
```

Using Python to Manage Users on Satellite 6

This example performs the same task as the previous example. That is, it uses a Python script to connect to and authenticate against a Satellite 6 Server, search for a specific user, delete that user if it exists, and then recreate it with Administrator privileges.

```
#!/usr/bin/python
import json
import requests

# Define Satellite location and login details
SAT_API = "https://localhost/api/v2/"
POST_HEADERS = {'content-type': 'application/json'}
USERNAME = "admin"
PASSWORD = "changeme"
SSL_VERIFY = False

def get_json(location):
    """
    Performs a GET using the passed URL location
    """
    r = requests.get(location, auth=(USERNAME, PASSWORD),
verify=SSL_VERIFY)

    return r.json()

def post_json(location, json_data):
    """
    Performs a POST and passes the data to the URL location
    """
    result = requests.post(
        location,
        data=json_data,
        auth=(USERNAME, PASSWORD),
        verify=SSL_VERIFY,
        headers=POST_HEADERS)

    return result.json()

def delete_json(location):
    """
    Performs a DELETE and passes the id to the URL location
    """
    result = requests.delete(
        location,
        auth=(USERNAME, PASSWORD),
        verify=SSL_VERIFY)

    return result.json()

def main():
    # List all users within the Satellite
    users = get_json(SAT_API + "users/")

    #print json.dumps(users, indent=4)
    print "Users known are:"
    for login in users['results']:
        print login['login']

    # Look for user example and if found, delete the user
    for delete in users['results']:
        if delete['login'] == 'example':
            id = delete ['id']
            id = str(id)

            delete = delete_json(SAT_API + "/users/" + id)
            #print json.dumps(delete, indent=4)
            print "User example deleted"

    # Create a user called example as admin role
    createuser = post_json(SAT_API + "/users/", json.dumps({ "mail":
"root@localhost", "firstname": "Example", "lastname": "User", "login":
"example", "password": "redhat", "admin": 'true', "auth_source_id": 1 }))
    #print json.dumps(createuser, indent=4)
    print "Admin user example created"

if __name__ == "__main__":
    main()
```

Using Ruby to Manage Users on Satellite 6

This example uses Ruby to perform the same task as the previous examples.

```
#!/usr/bin/ruby193-ruby
require 'json'
require 'rest-client'

url = 'https://localhost/api/v2/'
$username = 'admin'
$password = 'changeme'

def get_json(location)
    response = RestClient::Request.new(
        :method => :get,
        :url => location,
        :user => $username,
        :password => $password,
        :headers => { :accept => :json,
        :content_type => :json }
    ).execute
    results = JSON.parse(response.to_str)
end

def post_json(location, json_data)
    response = RestClient::Request.new(
        :method => :post,
        :url => location,
        :user => $username,
        :password => $password,
        :headers => { :accept => :json,
        :content_type => :json},
        :payload => json_data
    ).execute
    results = JSON.parse(response.to_str)
end

def delete_json(location)
    response = RestClient::Request.new(
        :method => :delete,
        :url => location,
        :user => $username,
        :password => $password,
        :headers => { :accept => :json,
        :content_type => :json }
    ).execute
    results = JSON.parse(response.to_str)
end

# List all users within the Satellite
users = get_json(url+"users/")

#puts JSON.pretty_generate(users)
puts "Users known are:"
users['results'].each do |name|
    puts name['login']
end

# Look for user example and if found, delete the user
users['results'].each do |name|
    if name['login'] == 'example'
        id = name['id']
        delete = delete_json(url+"users/"+id.to_s)
        #puts JSON.pretty_generate(delete)
        puts "User example deleted"
    end
end

# Create a user called example as admin role
data = JSON.generate({ :mail => "root@localhost", :firstname => "Example",
:lastname => "User", :login => "example", :password => "password", :admin
=> 'true', :auth_source_id => 1})
createuser = post_json(url+"users/", data)

#puts JSON.pretty_generate(createuser)
puts "Admin user example created"

exit()
```

Using the Command Line to Manage Users on Satellite 5

This example uses the **spacecmd** command to perform the same task as the previous examples.

```
# spacecmd -u admin -p password user_list
# spacecmd -u admin -p password user_delete example
# spacecmd -u admin -p password user_create
# spacecmd -u admin -p password user_addrole example org_admin
```

An example session might appear as follows:

```
# spacecmd -u admin -p password user_list
INFO: Spacewalk Username: admin
INFO: Connected to https://localhost/rpc/api as admin
admin
example

# spacecmd -u admin -p password user_delete example
INFO: Spacewalk Username: admin
INFO: Connected to https://localhost/rpc/api as admin

Delete this user [y/N]: y

# spacecmd -u admin -p password user_list
INFO: Spacewalk Username: admin
INFO: Connected to https://localhost/rpc/api as admin
admin

# spacecmd -u admin -p password user_create
INFO: Spacewalk Username: admin
INFO: Connected to https://localhost/rpc/api as admin
Username: example
First Name: Example
Last Name: User
Email: root@localhost
PAM Authentication [y/N]: n
Password:
```

```
Repeat Password:

# spacecmd -u admin -p password user_list
INFO: Spacewalk Username: admin
INFO: Connected to https://localhost/rpc/api as admin
admin
example

# spacecmd -u admin -p password user_addrole example org_admin
INFO: Spacewalk Username: admin
INFO: Connected to https://localhost/rpc/api as admin

# spacecmd -u admin -p password user_details example
INFO: Spacewalk Username: admin
INFO: Connected to https://localhost/rpc/api as admin
Username:      example
First Name:    Example
Last Name:     User
Email Address: root@localhost
Organization:  MY ORG
Last Login:
Created:       8/19/14 8:42:52 AM EDT
Enabled:       True

Roles
-----
activation_key_admin
channel_admin
config_admin
monitoring_admin
org_admin
system_group_admin
```

Using the Command Line to Manage Users on Satellite 6

This example uses the **hammer** command to perform the same task as the previous examples.

```
# hammer shell
> user list
> user delete --id 4 --login example
> user create --admin true --firstname Example --lastname User --login
example --mail root@localhost --password redhat --auth-source-id 1
```

An example session might appear as follows:

```
# hammer shell
[Foreman] password for admin:
Welcome to the hammer interactive shell
Type 'help' for usage information

hammer> user list
---|---------|--------------|---------------
ID | LOGIN   | NAME         | EMAIL
---|---------|--------------|---------------
4  | example | Example User | root@localhost
```

```
3   | admin    | Admin User    | root@localhost
---|---------|--------------|---------------

hammer> user delete --id 4 --login example
User deleted
hammer> user list
---|-------|------------|---------------
ID | LOGIN | NAME       | EMAIL
---|-------|------------|---------------
3  | admin | Admin User | root@localhost
---|-------|------------|---------------

hammer> user create --admin true --firstname Example --lastname User --
login example --mail root@localhost --password redhat --auth-source-id 1
User created
hammer> user list
---|----------|--------------|---------------
ID | LOGIN    | NAME         | EMAIL
---|----------|--------------|---------------
3  | admin    | Admin User   | root@localhost
5  | example  | Example User | root@localhost
---|----------|--------------|---------------
hammer>
```

5.2. SATELLITE 6 API TIPS

This section provides some general tips to help optimize your experience with the Satellite 6 API.

Browsing the Satellite 6 API

If you have already logged in to the Satellite 6 web UI, you can see the default results of GET requests at /api/v2/<API-NAME>. For example:

- https://satellite6.example.com/api/v2/users/
- https://satellite6.example.com/api/v2/users/3

Using Satellite 6 API Requests on the Command Line

You can use the **`curl`** command to interact with the Satellite 6 API. For example:

Example 5.1. Example GET Requests

Example GET requests to list organizations, hosts, and users within Satellite 6.

```
# SATUSER=admin
# SATPASS='changeme'
# SATURL="https://localhost"

# curl -k -u $SATUSER:$SATPASS -X GET -H \
'Accept: application/json' $SATURL/api/v2/organizations | json_reformat
# curl -k -u $SATUSER:$SATPASS -X GET -H \
'Accept: application/json' $SATURL/api/v2/hosts | json_reformat
# curl -k -u $SATUSER:$SATPASS -X GET -H \
'Accept: application/json' $SATURL/api/v2/users | json_reformat
# curl -k -u $SATUSER:$SATPASS -X GET -H \
'Accept: application/json' $SATURL/api/v2/users/3 | json_reformat
```

Example 5.2. Example DELETE Request

An example DELETE request to remove an existing user with known ID "9" based on a previous list of users.

```
# curl -k -u $SATUSER:$SATPASS -X DELETE -H \
'Accept: application/json' $SATURL/api/v2/users/9 | json_reformat
```

Example 5.3. Example POST Request

An example POST request to create a new user called **example**, passing the `true` flag to enable administrator privileges for the user.

```
# curl -k -u $SATUSER:$SATPASS -X POST \
-d '{ "mail": "root@localhost", "firstname": "Example", \
"lastname": "User", "login": "example", "password": "redhat", \
"admin": 'true', "auth_source_id": 1 }' \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
$SATURL/api/v2/users | json_reformat
```

Example 5.4. Example PUT Request

An example PUT request to change the email address of the existing **example** user with known ID "10" to *example@localhost*.

```
# curl -k -u $SATUSER:$SATPASS -X PUT \
-d '{ "id": 10, "mail": "example@localhost" }' \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
$SATURL/api/v2/users/10 | json_reformat
```

APPENDIX A. GLOSSARY OF TERMS

The following terms are used throughout this document. Familiarize yourself with these terms to help your understanding of Red Hat Satellite 6.

Activation Key

A registration token used in a Kickstart file to control actions at registration. These are similar to Activation Keys in Red Hat Satellite 5, but provide a subset of features because Puppet controls package and configuration management after registration.

Application Life Cycle Environment

An Application Life Cycle Environment represents a step, or stage, in a promotion path through the Software Development Life Cycle (SDLC). Promotion paths are also known as development paths. Content such as packages and Puppet modules move through life cycle environments by publishing and promoting Content Views. All Content Views have versions, which means you can promote a specific version through a typical promotion path; for example, from development to test to production. Channel cloning implements this concept in Red Hat Satellite 5.

Attach

The process of associating a Subscription to a Host that provides access to RPM content.

Capsule

A Capsule is an additional server that can be used in a Red Hat Satellite 6 deployment to facilitate content federation and distribution in addition to other localized services (Puppet Master, **DHCP**, **DNS**, **TFTP**, and more).

Catalog

A Catalog is a document that describes the desired system state for one specific computer. It lists all of the resources that need to be managed, as well as any dependencies between those resources.

Compute Profile

Compute Profiles specify default attributes for new virtual machines on a compute resource.

Compute Resource

A Compute Resource is virtual or cloud infrastructure, which Red Hat Satellite 6 uses for deployment of hosts and systems. Examples include Red Hat Enterprise Virtualization Manager, OpenStack, EC2, and VMware.

Content

Content includes software packages (RPM files) and Puppet modules. These are synchronized into the Library and then promoted into Life Cycle Environments using Content Views so that they can be consumed by Hosts.

Content Delivery Network (CDN)

The Content Delivery Network (CDN) is the mechanism used to deliver Red Hat content in a geographically co-located fashion. For example, content that is synchronized by a Satellite in Europe pulls content from a source in Europe.

Content Host

A Content Host is the part of a host that manages tasks related to content and subscriptions.

Content View

A Content View is a definition of content that combines Products, packages, and Puppet modules with capabilities for intelligent filtering and creating snapshots. Content Views are a refinement of the combination of channels and cloning from Red Hat Satellite 5.

External Node Classifier

An External Node Classifier is a Puppet construct that provides additional data for a Puppet Master to use when configuring Hosts. Red Hat Satellite 6 acts as an External Node Classifier to Puppet Masters in a Satellite deployment.

Facter

Facter is a program that provides information (facts) about the system on which it is run; for example, Facter can report total memory, operating system version, architecture, and more. Puppet modules enable specific configurations based on host data gathered by Facter.

Hammer

Hammer is a command line tool for Red Hat Satellite 6. Use Hammer to manage Red Hat Satellite 6 as a standard CLI, for scripts, and also through an interactive shell.

Hiera

Hiera is a key-value look-up tool for configuration data which allows keeping site-specific data out of Puppet manifests.

Host

A Host refers to any system, either physical or virtual, that Red Hat Satellite 6 manages.

Host Collection

A Host Collection is equivalent to a Satellite 5 System Group, that is, a user defined group of one or more Hosts.

Host Group

A Host Group is a template for building a Host. This includes the content view (which defines the available RPM files and Puppet modules) and the Puppet classes to apply (which ultimately determines the software and configuration).

Location

A Location is collection of default settings that represent a physical place. These can be nested so that you can set up an hierarchical collection of locations. For example, you can set up defaults for "Middle East", which are refined by "Tel Aviv", which are further refined by "Data Center East", and then finally by "Rack 22".

Library

The Library contains **every** version, including the latest synchronized version, of the software that the user will ever deploy. For an Information Technology Infrastructure Library (ITIL) organization or department, this is the Definitive Media Library (previously named the Definitive Software Library).

Manifest

A manifest transfers subscriptions from the Customer Portal to Red Hat Satellite 6. This is similar in function to certificates used with Red Hat Satellite 5.
For more information about certificates and subscription types, see:

- Subscription Concepts and Workflows.

- Migrating from RHN Classic.

Organization

An Organization is an isolated collection of systems, content, and other functionality within a Satellite 6 deployment.

Product

A collection of content repositories. Products can be Red Hat Products or newly-created Products made up of software and configuration content.

Promote

The act of moving a content view comprised of software and configuration content from one Application Life Cycle Environment to another, such as moving from development to QA to production.

Provisioning Template

A Provisioning Template is a user-defined template for Kickstart files, snippets, and other provisioning actions. In Satellite 6 they provide similar functionality to Kickstart Profiles and cobbler Snippets in Red Hat Satellite 5.

Pulp Node

A Pulp Node is a Capsule Server component that mirrors content. This is similar to the Red Hat Satellite 5 Proxy. The main difference is that content can be staged on the Pulp Node before it is used by a Host.

Puppet Agent

The Puppet Agent is an agent that runs on a Host and applies configuration changes to that Host.

Puppet Master

A Puppet Master is a Capsule Server component that provides Puppet manifests to Hosts for execution by the Puppet Agent.

Puppet Module

A Puppet module is a self-contained bundle of code and data that you can use to manage resources such as users, files, and services.

Repository

A Repository provides storage for a collection of content. For example, a yum repository or a Puppet repository.

Role

A Role specifies a collection of permissions that are applied to a set of resources, such as Hosts.

Smart Proxy

A Smart Proxy is a Capsule Server component that can integrate with external services, such as **DNS** or **DHCP**.

Smart Variable

A Smart Variable is a configuration value that controls how a Puppet Class behaves. This can be set on a Host, a Host Group, an Organization, or a Location.

Standard Operating Environment (SOE)

A Standard Operating Environment (SOE) is a controlled version of the operating system on which applications are deployed.

Subscription

Subscriptions are the means by which you receive content and service from Red Hat.

Synchronizing

Synchronizing refers to mirroring content from external resources into the Red Hat Satellite 6 Library.

Synchronization Plans

Synchronization Plans provide scheduled execution of content synchronization.

User Group

A User Group is a collection of roles which can be assigned to a collection of users. This is similar to a Role in Red Hat Satellite 5.

User

A user is anyone registered to use Red Hat Satellite. Authentication and authorization is possible through built-in logic, through external LDAP resources, or with Kerberos.