



# Red Hat Satellite 6.4

## Content Management Guide

An end-to-end guide on managing content from Red Hat and custom sources



# Red Hat Satellite 6.4 Content Management Guide

---

An end-to-end guide on managing content from Red Hat and custom sources

Red Hat Satellite Documentation Team

[satellite-doc-list@redhat.com](mailto:satellite-doc-list@redhat.com)

## Legal Notice

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

This guide provides an end-to-end scenario on managing content in Red Hat Satellite 6. Examples of such content include RPM files, ISO images, Puppet modules, and container images. Red Hat Satellite 6 manages this content using a set of Content Views promoted across the application lifecycle. This guide demonstrates how to create an application lifecycle that suits your organization and content views that fulfils host states within lifecycle environments. These content views eventually form the basis for provisioning and updating hosts in your Red Hat Satellite 6 environment.

# Table of Contents

<b>CHAPTER 1. INTRODUCTION</b> .....	<b>5</b>
1.1. OVERVIEW OF RED HAT SATELLITE 6 CONTENT MANAGEMENT	5
1.2. DEFINING CONTENT MANAGEMENT TYPES	5
1.3. DEFINING OUR SCENARIO	6
1.4. CONTENT MANAGEMENT STORAGE	6
<b>CHAPTER 2. MANAGING ORGANIZATIONS</b> .....	<b>9</b>
2.1. CREATING AN ORGANIZATION	9
2.2. SETTING THE ORGANIZATION CONTEXT	10
2.3. CREATING AN ORGANIZATION DEBUG CERTIFICATE	10
2.4. USING AN ORGANIZATION DEBUG CERTIFICATE	11
2.5. DELETING AN ORGANIZATION	12
<b>CHAPTER 3. MANAGING LOCATIONS</b> .....	<b>13</b>
3.1. CREATING A LOCATION	13
3.2. SETTING THE LOCATION CONTEXT	13
3.3. DELETING A LOCATION	14
<b>CHAPTER 4. MANAGING SUBSCRIPTIONS</b> .....	<b>15</b>
4.1. MANAGING SUBSCRIPTION ALLOCATIONS	15
4.1.1. Creating a Subscription Allocation in the Customer Portal	15
4.1.2. Adding Subscriptions to a Subscription Allocation	15
4.1.3. Removing Subscriptions from a Subscription Allocation	16
4.1.4. Moving Subscriptions between Subscription Allocations	16
4.1.5. Using Future Dated subscriptions	17
4.1.6. Using Subscription Allocations to Manage Multiple Organizations	17
4.1.7. Exporting a Subscription Manifest from the Customer Portal	17
4.1.8. Importing a Subscription Manifest into Satellite Server	17
4.2. MANAGING SUBSCRIPTION ALLOCATIONS IN THE SATELLITE WEB UI	18
4.2.1. Searching for a Subscription in the Satellite Web UI	18
4.2.2. Attaching Subscriptions to Content Hosts	19
4.2.3. Adding Subscriptions to Subscription Allocations in the Satellite Web UI	20
4.2.4. Removing Subscriptions from Subscription Allocations in the Satellite Web UI	20
4.2.5. Updating and Refreshing Subscription Manifests	21
4.2.6. Bulk Updating Content Hosts' Subscriptions	21
<b>CHAPTER 5. IMPORTING RED HAT CONTENT</b> .....	<b>23</b>
5.1. CREATING A DEFINITIVE MEDIA LIBRARY	23
5.2. USING PRODUCTS AND REPOSITORIES IN SATELLITE	23
5.3. SYNCHRONIZING CONTENT	23
5.3.1. Disabling the Global HTTP Proxy	23
5.4. CONFIGURING DOWNLOAD POLICIES	24
5.4.1. Changing the Default Download Policy	25
5.4.2. Changing the Download Policy for a Repository	25
5.5. SELECTING RED HAT REPOSITORIES TO SYNCHRONIZE	25
5.6. SYNCHRONIZING RED HAT REPOSITORIES	27
5.6.1. Recovering a Repository	29
5.6.2. Limiting Synchronization Speed	30
5.7. CREATING A SYNCHRONIZATION PLAN	30
<b>CHAPTER 6. IMPORTING CUSTOM CONTENT</b> .....	<b>32</b>
6.1. USING CUSTOM PRODUCTS IN SATELLITE	32

6.2. IMPORTING CUSTOM SSL CERTIFICATES	32
6.3. IMPORTING A CUSTOM GPG KEY	32
6.4. CREATING A CUSTOM PRODUCT	34
6.5. ADDING A CUSTOM RPM REPOSITORY	34
<b>CHAPTER 7. MANAGING APPLICATION LIFE CYCLES</b>	<b>36</b>
7.1. APPLICATION LIFE CYCLE OVERVIEW	36
7.2. PROMOTING CONTENT ACROSS THE APPLICATION LIFE CYCLE	37
7.3. CREATING A LIFE CYCLE ENVIRONMENT PATH	38
7.4. REMOVING LIFE CYCLE ENVIRONMENTS FROM SATELLITE SERVER	39
7.5. REMOVING LIFE CYCLE ENVIRONMENTS FROM CAPSULE SERVER	40
7.6. ADDING LIFE CYCLE ENVIRONMENTS TO CAPSULE SERVERS	41
<b>CHAPTER 8. MANAGING CONTENT VIEWS</b>	<b>43</b>
8.1. CREATING A CONTENT VIEW	44
8.2. CREATING A CONTENT VIEW WITH A PUPPET MODULE	45
8.3. PROMOTING A CONTENT VIEW	45
8.4. COMPOSITE CONTENT VIEWS OVERVIEW	46
8.5. CREATING A COMPOSITE CONTENT VIEW	48
8.6. CONTENT FILTER OVERVIEW	49
8.7. CONTENT FILTER EXAMPLES	50
8.8. CREATING A CONTENT FILTER	51
<b>CHAPTER 9. MANAGING ACTIVATION KEYS</b>	<b>53</b>
9.1. CREATING AN ACTIVATION KEY	53
9.2. UPDATING SUBSCRIPTIONS ASSOCIATED WITH AN ACTIVATION KEY	55
9.3. UPDATING SUBSCRIPTIONS ASSOCIATED WITH AN ACTIVATION KEY USING A CSV FILE	56
9.4. USING ACTIVATION KEYS FOR HOST REGISTRATION	58
9.5. ENABLING AUTO-ATTACH	59
9.6. SETTING THE SERVICE LEVEL	60
<b>CHAPTER 10. MANAGING ERRATA</b>	<b>61</b>
10.1. INSPECTING AVAILABLE ERRATA	61
10.2. SUBSCRIBING TO ERRATA NOTIFICATIONS	63
10.3. MANAGING ERRATA WITH CONTENT VIEWS	63
10.3.1. Creating a Content View Filter for Errata	63
10.4. APPLYING ERRATA TO INDIVIDUAL SYSTEMS	65
10.5. APPLYING ERRATA TO MULTIPLE SYSTEMS	66
<b>CHAPTER 11. MANAGING OSTREE CONTENT</b>	<b>68</b>
11.1. SELECTING RED HAT OSTREE CONTENT TO SYNCHRONIZE	68
11.2. IMPORTING CUSTOM OSTREE CONTENT	69
11.3. MANAGING OSTREE CONTENT WITH CONTENT VIEWS	70
<b>CHAPTER 12. MANAGING ISO IMAGES</b>	<b>72</b>
12.1. IMPORTING ISO IMAGES FROM RED HAT	72
12.2. IMPORTING INDIVIDUAL ISO IMAGES AND FILES	73
<b>CHAPTER 13. MANAGING CUSTOM FILE TYPE CONTENT</b>	<b>75</b>
13.1. CREATING A CUSTOM FILE TYPE REPOSITORY IN RED HAT SATELLITE	75
13.2. CREATING A CUSTOM FILE TYPE REPOSITORY IN A LOCAL DIRECTORY	77
13.3. CREATING A REMOTE FILE TYPE REPOSITORY	79
13.4. UPLOADING FILES TO A CUSTOM FILE TYPE REPOSITORY IN RED HAT SATELLITE	80
13.5. DOWNLOADING FILES TO A HOST FROM A CUSTOM FILE TYPE REPOSITORY IN RED HAT SATELLITE	81

---

<b>CHAPTER 14. MANAGING CUSTOM PUPPET CONTENT</b> .....	<b>83</b>
14.1. CREATING A CUSTOM PUPPET REPOSITORY	83
14.2. MANAGING INDIVIDUAL PUPPET MODULES	83
14.3. SYNCHRONIZING PUPPET REPOSITORIES	84
14.4. SYNCHRONIZING PUPPET MODULES FROM A GIT REPOSITORY	85
<b>CHAPTER 15. SYNCHRONIZING CONTENT BETWEEN SATELLITE SERVERS</b> .....	<b>87</b>
15.1. CONFIGURING AN EXPORT DESTINATION	87
15.2. SYNCHRONIZING CONTENT USING CONTENT VIEW VERSIONS	88
15.2.1. Creating a Content View for Exporting	88
15.2.2. Exporting a Content View Version	90
15.2.3. Importing a Content View as a Red Hat Repository	92
15.3. SYNCHRONIZING CONTENT USING REPOSITORIES	93
15.3.1. Exporting a repository	93
15.3.2. Incremental Export of a Repository	94
15.3.3. Importing a Repository	94
15.3.4. Incremental Import of a Repository	95
<b>APPENDIX A. USING AN NFS SHARE FOR CONTENT STORAGE</b> .....	<b>96</b>
<b>APPENDIX B. IMPORTING CONTENT ISOS INTO A DISCONNECTED SATELLITE</b> .....	<b>98</b>
<b>APPENDIX C. IMPORTING CONTENT ISOS INTO A CONNECTED SATELLITE</b> .....	<b>100</b>
<b>APPENDIX D. SYNCHRONIZING TEMPLATES WITH GIT</b> .....	<b>103</b>
D.1. ENABLING THE TEMPLATESYNC PLUG-IN	103
D.2. CONFIGURING THE TEMPLATESYNC PLUG-IN	103
D.3. IMPORTING AND EXPORTING TEMPLATES	105
D.3.1. Synchronizing Templates with a Git repository	105
D.3.2. Synchronizing templates with a local directory	106
D.4. ADVANCED GIT CONFIGURATION	107
D.5. UNINSTALLING THE PLUG-IN	107





# CHAPTER 1. INTRODUCTION

In the context of system management, we define **content** as the software installed on systems. This includes, but is not limited to, the base operating system, middleware services, and end user applications. Red Hat Satellite 6 provides tools to manage the various types of content for Red Hat Enterprise Linux systems. This provides system administrators with an easy way to collect a range of content, keep it up to date, and use it to provision new systems and update existing systems.

This guide aims to provide an end-to-end scenario to demonstrate how to manage your content. This guide is targeted at system administrators with a newly installed Satellite Server.

## 1.1. OVERVIEW OF RED HAT SATELLITE 6 CONTENT MANAGEMENT

Content management in the context of Red Hat Satellite 6 means a work flow that provides a sustainable repository for multiple content types. For Red Hat content, Red Hat Satellite 6 also uses subscription information so that it knows what content is available to a user. This means Red Hat Satellite 6 uses components to manage the following:

- **Subscription management**, which includes tools to manage Red Hat software subscriptions, and associated content, over a secure connection. This provides a means for organizations to manage their Red Hat subscription information.
- **Content management**, which includes applications to download and store content in custom repositories. This provides organizations with a method to store Red Hat content and organize it in various ways.

## 1.2. DEFINING CONTENT MANAGEMENT TYPES

Red Hat Satellite 6 manages different content types. This includes:

### RPM Packages

Red Hat Satellite 6 provides a method to import RPM files from repositories related to your Red Hat subscriptions. This means Satellite Server downloads the RPM files from Red Hat's Content Delivery Network and stores them locally. You can use these repositories and their RPM files in Content Views.

### Kickstart Trees

Red Hat Satellite 6 obtains the kickstart trees for creating a new system. New systems access these kickstart trees over a network to use as base content for their installation. Red Hat Satellite 6 also contains some predefined kickstart templates (and the ability to create your own), which are used to provision new systems and customize the installation.

### ISO and KVM Images

Red Hat Satellite 6 downloads and manages media for installation and provisioning. For example, Satellite downloads, stores and manages ISO images and KVM guest images for specific Red Hat Enterprise Linux versions.

### Puppet Modules

Red Hat Satellite 6 offers the ability to upload Puppet modules alongside RPM content so that it can configure the system's state after provisioning. Users can also manage Puppet classes and parameters as part of the provisioning process.

### Container Images

Red Hat Satellite 6 can act as a registry for container images. This provides a method to create containers using Red Hat Enterprise Linux Atomic Host.

## OSTree

Red Hat Satellite 6 can import OSTree branches and publish this content to a HTTP location.

### 1.3. DEFINING OUR SCENARIO

This guide uses an example scenario to demonstrate Red Hat Satellite 6's features. In this scenario, a software development company called **ACME** recently installed Red Hat Satellite 6 and its system administrators want to start importing and managing Red Hat content. The end goal for ACME is to:

- Have a set of content sources imported for their organization. This includes content from their Red Hat subscription and their own custom content.
- Have an application life cycle defined based on their software development process.
- Have their Satellite Server ready so they can provision new Red Hat Enterprise Linux hosts and register existing Red Hat Enterprise Linux hosts.

This guide only focuses on content management. Further features, such as host provisioning, environment architecture, and Satellite Server management, are covered in other guides in the Red Hat Satellite 6 series.

This guide provides both steps for using either the Red Hat Satellite 6 Web UI or its CLI tool – Hammer CLI. Use either depending on your preferred method of interacting with Red Hat Satellite 6. If using the CLI and do not want to include authentication details each time you enter a **hammer** command, edit the `~/.hammer/cli.modules.d/foreman.yml` file to include the credentials for a local user:

```
:foreman:  
  :host: 'https://satellite.example.com/'  
  :username: 'your_username'  
  :password: 'your_password'
```



#### IMPORTANT

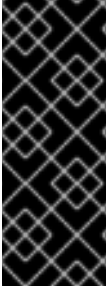
All uses of the **hammer** command in this guide use the configuration file and omit the authentication details.

Some CLI commands can use the **--async** option to perform certain tasks asynchronously. For example, you can synchronize content and publish Content Views as a asynchronous task rather than monitoring it. This guide omits the **--async** so that users can monitor progress of tasks to completion. If including the **--async** option for certain tasks, ensure the task completes before moving on to the next task.

### 1.4. CONTENT MANAGEMENT STORAGE

Red Hat Satellite 6 hosts repositories for RPM and Puppet content, including content synchronized with Red Hat's Content Delivery Network and your own custom repositories. Such repositories grow in size over time. This means you must estimate adequate size requirements to suit your environment and scale future requirements accordingly.

Red Hat Satellite 6 temporarily adds content to `/var/cache/pulp`, then moves them into the `/var/lib/pulp` directory for storage and management. Ensure that both `/var/cache/pulp` and `/var/lib/pulp` have adequate storage. Mount the `/var/lib/pulp` directory onto a large local partition that you can scale. For example, use Logical Volume Manager (LVM) to create this partition.



## IMPORTANT

Do not mount **/var/lib/pulp** on an NFS share. Parts of Red Hat Satellite 6 use transient SQLite databases, which have issues over NFS. If you want to use an NFS share, mount only the **/var/lib/pulp/content** directory, which contains the main source content units. Use of high-bandwidth, low-latency storage for the **/var/lib/pulp** file system. Red Hat Satellite has many operations that are I/O-intensive so usage of high-latency, low-bandwidth storage could result in performance degradation.

Red Hat Satellite 6 synchronize and stores packages from Red Hat's Content Delivery Network. This includes repositories for Red Hat Enterprise Linux and other Red Hat software. The recommended storage requirements for Red Hat content are as follows:

### During Production Phase 1 of a major Red Hat Enterprise Linux version:

- At least 40GB for each binary package repository
- At least 80GB for each debug-info repository

### After Production Phase 1 of a major Red Hat Enterprise Linux version:

- The estimated annual growth rates of these repositories are 10GB per binary package repository and 20GB per debug-info repository

For more information about Red Hat Production Phases, see ["Red Hat Enterprise Linux Life Cycle"](#) .



## NOTE

All repositories vary in size. These specifications are recommendations and must be adjusted according to the repositories you want to synchronize.

Red Hat Satellite 6 provides users with the ability to create Content Views. Content Views act as a snapshot of a user-defined content collection at a particular point in time. This provides a means to create customized content collections from preexisting repositories.

Each unit of content in a Content View uses a symbolic link to the Definitive Media Library stored in the **/var/lib/pulp/content** directory. In addition, each repository in a Content View contains metadata about the content belonging to the Content View. This means a Content View using a minimal number of packages uses a small amount of storage. However, the storage size adds up when you use multiple Content Views and a large number of packages per view.

For example, a Content View using the Red Hat Enterprise Linux 7 RPMs repository might contain over 7000 packages. In terms of disk space, this might only result in less than 100MB of symbolic links. However, take into account the following:

- The number of Content Views containing this repository
- The number of versions per Content View
- The number of life cycle environments using the promoted view
- Any additional content, such as kickstart trees or Live CD content

To help reduce the amount of storage that Content Views consume, use the following recommendations:

- Remove unused versions of Content Views. If you are not using a Content View in a life cycle environment and you have no intention of reusing it, delete it to reclaim the storage used.
- Use filters in Content Views. Filters limit the content that appears in a Content View. This helps you define only the necessary content required for your view and excludes redundant content. This reduces the size of each Content View significantly.
- Monitor the **/var/lib/pulp/nodes** directory. Red Hat Satellite 6 uses this directory to construct Content Views.
- Monitor the **/var/lib/pulp/published** directory. Red Hat Satellite 6 uses this directory to publish Content Views.

Red Hat Satellite 6 also uses the following databases for its content management components:

- **Main database** - PostgreSQL database stored in **/var/lib/pgsql**. The storage requirements depend on a number of factors including organizations, environments, registered systems, and Content Views.
- **Content database** - MongoDB database stored in **/var/lib/mongodb**. The storage requirements depend on the number of packages and Content Views for your Red Hat Satellite 6 environment. This usually takes a large amount of storage. Reserve at minimum 10GB for the Content database and plan for an estimated 5GB per repository. It is recommended to mount this directory onto a large local partition that you can scale. For example, use Logical Volume Manager (LVM) to create this partition.



### IMPORTANT

Do not mount **/var/lib/mongodb** on an NFS share. Red Hat recommends the usage of high-bandwidth, low-latency storage for the **/var/lib/mongodb** file system. Red Hat Satellite has many operations that are I/O-intensive so usage of high-latency, low-bandwidth storage could result in performance degradation.

## CHAPTER 2. MANAGING ORGANIZATIONS

Organizations divide Red Hat Satellite 6 resources into logical groups based on ownership, purpose, content, security level, or other divisions. You can create and manage multiple organizations through Red Hat Satellite 6, then divide and assign your Red Hat subscriptions to each individual organization. This provides a method of managing the content of several individual organizations under one management system. Here are some examples of organization management:

### Single Organization

A small business with a simple system administration chain. In this case, we create a single organization for the business and assign content to it.

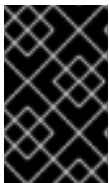
### Multiple Organizations

A large company that owns several smaller business units. For example, a company with separate system administration and software development groups. In this case, we create organizations for the company and each of the business units it owns. This keeps the system infrastructure for each separate. We then assign content to each organization based on their needs.

### External Organizations

A company that manages external systems for other organizations. For example, a company offering cloud computing and web hosting resources to customers. In this case, we might create an organization for the company's own system infrastructure and then an organization for each external business. We then assign content to each organization where necessary.

A default installation of Red Hat Satellite 6 provides a default organization called **Default\_Organization**.



### IMPORTANT

If a new user is not assigned a default organization their access is limited. To grant systems rights to users, assign them to a default organization and have them log out and log back in again.

## 2.1. CREATING AN ORGANIZATION

### For Web UI Users

1. Navigate to **Administer > Organizations**.
2. Click **New Organization**.
3. Provide the base details for the organization:
  - **Name** - A plain text name for the organization.
  - **Label** - A unique identifier for the organization. This is used for creating and mapping certain assets, such as directories for content storage. Use letters, numbers, underscores, and dashes, but no spaces.
  - **Description** - An optional plain text description for the organization.
4. Click **Submit**.
  - If you have hosts with no organization assigned, you are redirected to the **Select Hosts** tab. Select between three variants of assigning hosts to the organization:

- **Assign All** to assign all hosts with no organization assigned.
  - **Manually Assign** to select which hosts to assign to the organization.
  - **Proceed to Edit** to move to the **Edit Properties** page without assigning hosts to the organization
- If you do not have hosts with no organization assigned, you are redirected to the **Edit** page, where you can assign infrastructure resources to the organization. This includes networking resources, installation media, kickstart templates, and other parameters. You can return to this page at any time by navigating to **Administer > Organizations** and then selecting an organization to edit.
5. After completing your organization creation, click **Submit**.

### For CLI Users

Enter the following command to create an organization:

```
# hammer organization create \  
--name "your_organization_name" \  
--label "your_organization_label" \  
--description "your_organization_description"
```

## 2.2. SETTING THE ORGANIZATION CONTEXT

An organization context defines the organization to use for a host and its associated resources.

### For Web UI Users

The organization menu is the first menu item in the menu bar, on the upper left of the Satellite web UI. If you have not selected a current organization, the menu says **Any Organization**. Click the **Any Organization** button and select the organization to use.

### For CLI Users

While using the CLI, include either **--organization "your\_organization\_name"** or **--organization-label "your\_organization\_label"** as an option. For example:

```
# hammer subscription list --organization "Default_Organization"
```

This command outputs subscriptions allocated for the Default\_Organization.

## 2.3. CREATING AN ORGANIZATION DEBUG CERTIFICATE

### To Create a New Organization Debug Certificate:

1. Navigate to **Administer > Organizations**.
2. Select an organization for which you want to generate a debug certificate.
3. Click **Generate and Download**. This generates a debug certificate.
4. Save the certificate file in a secure location.

**NOTE**

Debug Certificates are automatically generated for provisioning template downloads if they do not already exist in the organization for which they are being downloaded.

## 2.4. USING AN ORGANIZATION DEBUG CERTIFICATE

You can view an organization's repository content using a browser or using the API if you have a debug certificate for that organization. The previous section describes creating and downloading the certificate which is in the X.509 format. To use a browser you must first convert the X.509 certificate to a format your browser supports and then import the certificate. The **curl** utility only requires extracting the certificate and key into separate files.

### To Use an Organization Debug Certificate in Firefox.

1. Create and download an organization certificate as described in [Section 2.3, "Creating an Organization Debug Certificate"](#).

2. Open the X.509 certificate, for example, for the default organization:

```
$ vi 'Default Organization-key-cert.pem'
```

3. Copy the contents of the file from: **-----BEGIN RSA PRIVATE KEY-----** to **-----END RSA PRIVATE KEY-----** inclusive, into a file called **key.pem**.
4. Copy the contents of the file from **-----BEGIN CERTIFICATE-----** to **-----END CERTIFICATE-----** inclusive, into a file called **cert.pem**.
5. Enter a command as follows to create a PKCS12 format certificate and enter a password or phrase when prompted:

```
$ openssl pkcs12 -keypbe PBE-SHA1-3DES -certpbe PBE-SHA1-3DES -export -in cert.pem
-inkey key.pem -out organization_label.pfx -name organization_name
Enter Export Password:
Verifying - Enter Export Password:
```

6. Using the preferences tab, import the resulting **pfx** file into your browser: Navigate to **Edit > Preferences > Advanced Tab**. Select **View Certificates** in the **Certificates** view to open the **Certificate Manager**. On the **Your Certificates** tab, click **Import** and select the **pfx** file to load. You are prompted for the password or phrase used when making the certificate.
7. Enter a URL in the following format into your browser's address bar to begin browsing for repositories:

```
http://satellite.example.com/pulp/repos/organization_label
```

Pulp uses the organization label so the URL must use the organization label too.

### To Use an Organization Debug Certificate with Curl:

1. Create and download an organization certificate as described in [Section 2.3, "Creating an Organization Debug Certificate"](#)
2. Open the X.509 certificate, for example, for the default organization:

■

```
$ vi 'Default Organization-key-cert.pem'
```

3. Copy the contents of the file from: **-----BEGIN RSA PRIVATE KEY-----** to **-----END RSA PRIVATE KEY-----** inclusive, into a file called **key.pem**.
4. Copy the contents of the file from **-----BEGIN CERTIFICATE-----** to **-----END CERTIFICATE-----** inclusive, into a file called **cert.pem**.
5. Find a valid URL for a repository. You can use the browsing method described in the previous procedure or use the web UI. For example, using the web UI, navigate to **Content > Products** and select a Product by name. On the **Repositories** tab, select a repository by name and look for the **Published At** entry.
6. To use **curl** to access a repository, enter the following command:

```
$ curl -k --cert cert.pem --key key.pem  
http://satellite.example.com/pulp/repos/Default_Organization/Library/content/dist/rhel/server/7/  
7Server/x86_64/sat-tools/6.4/os/
```

Ensure the paths to **cert.pem** and **key.pem** are the correct absolute paths otherwise the command fails silently.

## 2.5. DELETING AN ORGANIZATION

You can delete an organization if the organization is not associated with any life cycle environments or host groups. If there are any life cycle environments or host groups associated with the organization you are about to delete, remove them by navigating to **Administer > Organizations** and clicking the relevant organization. It is not recommended to delete the default organization created during installation because the default organization is a placeholder for any unassociated hosts in the Satellite environment. There must be at least one organization in the environment in any given time.

### For Web UI Users

1. Navigate to **Administer > Organizations**.
2. Select **Delete** from the list to the right of the name of the organization you want to delete.
3. Click **OK** to delete the organization.

### For CLI Users

Enter the following command to delete an organization:

```
# hammer organization delete --organization "your_organization_name"
```



## CHAPTER 3. MANAGING LOCATIONS

Locations function similar to organizations, they provide a method to group resources and assign hosts. Differences between organizations and locations are:

- Locations are based on physical or geographical settings.
- Locations have hierarchical structure.

### 3.1. CREATING A LOCATION

#### For Web UI Users

1. Navigate to **Administer > Locations**
2. Click **New Location**.
  - Provide the base details for the location:
    - **Parent** - An optional parent location for this location. This creates a location hierarchy.
    - **Name** - A plain text name for the location.
    - **Description** - An optional plain text description for the location.
3. Click **Submit**.
  - If you have hosts with no location assigned, you are redirected to the **Select Hosts** tab. Select between three variants of assigning hosts to the location:
    - **Assign All** to assign all hosts with no location assigned.
    - **Manually Assign** to select which hosts to assign to the location.
    - **Proceed to Edit** to move to the **Edit Properties** page without assigning hosts to the location.
  - If you do not have hosts with no location assigned, you are redirected to the **Edit** page, where you can assign infrastructure resources to the location. This includes networking resources, installation media, kickstart templates, and other parameters. You can return to this page at any time by navigating to **Administer > Locations** and then selecting a location to edit.
4. After completing your location creation, click **Submit**.

#### For CLI Users

Enter the following command to create a location:

```
# hammer location create \  
--parent-id "parent_location_id" \  
--name "your_location_name" \  
--description "your_location_description"
```

### 3.2. SETTING THE LOCATION CONTEXT

A location context defines the location to use for a host and its associated resources.

### For Web UI Users

The location menu is the second menu item in the menu bar, on the upper left of the Satellite web UI. If you have not selected a current location, the menu says **Any Location**. Click the **Any location** button and select the location to use.

### For CLI Users

While using the CLI, include either `--location "your_location_name"` or `--location-id "your_location_id"` as an option. For example:

```
# hammer subscription list --location "Default_Location"
```

This command outputs subscriptions allocated for the Default\_Location.

## 3.3. DELETING A LOCATION

You can delete a location if the location is not associated with any life cycle environments or host groups. If there are any life cycle environments or host groups associated with the location you are about to delete, remove them by navigating to **Administer > Locations** and clicking the relevant location. It is not recommended to delete the default location created during installation because the default location is a placeholder for any unassociated hosts in the Satellite environment. There must be at least one location in the environment in any given time.

### For Web UI Users

1. Navigate to **Administer > Locations**.
2. Select **Delete** from the list to the right of the name of the location you want to delete.
3. Click **OK** to delete the location.

### For CLI Users

Enter the following command to delete a location:

```
# hammer location delete --location "your_location_name"
```

## CHAPTER 4. MANAGING SUBSCRIPTIONS

Red Hat Satellite 6 imports content from the Red Hat Content Delivery Network (CDN). A subscription manifest is used to find, access, and download content from the corresponding repositories. You must have a subscription manifest containing a subscription allocation for each organization on your Satellite Server. All subscription information is available in your Red Hat Customer Portal account.

Use this chapter to create, modify, and refresh subscription allocations and manifests.

### 4.1. MANAGING SUBSCRIPTION ALLOCATIONS

#### 4.1.1. Creating a Subscription Allocation in the Customer Portal

You can access your subscription information in the Red Hat Customer Portal and create a subscription allocation. You can then export the allocation as a subscription manifest for use in on-premise management applications, such as Red Hat Satellite.

##### Procedure

1. Open the Red Hat Customer Portal in your browser and log in to your Red Hat account.
2. Navigate to **Subscriptions**, which is in the upper-left corner of the Customer Portal page.
3. Click the **Subscription Allocations** tab.
4. Click **New Subscription Allocation**.
5. In the **Name** field, enter a descriptive name that helps identify the subscription allocation.
6. From the **Type** list, select the type and version that corresponds to your Satellite Server.
7. Click **Create**.
8. After a few minutes, the web UI reports that the allocation has been successfully created.

#### 4.1.2. Adding Subscriptions to a Subscription Allocation

If you want to add additional Red Hat products or future-dated subscriptions for existing products, you must add the relevant subscriptions to your subscription allocation. The following procedure explains how to add subscriptions to a subscription allocation.

##### Procedure

1. Open the Red Hat Customer Portal in your browser and log in to your Red Hat account.
2. Navigate to **Subscriptions**, which is in the upper-left corner of the Customer Portal page.
3. Click the **Subscription Allocations** tab.
4. From the **Name** column, select the name of the subscription allocation you want to modify.
5. Click the **Subscriptions** tab, and then click **Add Subscriptions**.
6. On the row of each subscription you want to add, enter the quantity under the **Entitlements** column.

7. Click **Submit**.
8. If the modified subscription allocation has a corresponding subscription manifest on a Satellite Server, refresh the subscription manifest. For more information, see [Section 4.2.5, "Updating and Refreshing Subscription Manifests"](#).

### 4.1.3. Removing Subscriptions from a Subscription Allocation

The following procedure explains how to remove subscriptions from a subscription allocation, for example, if you want to remove expired subscriptions.

#### Procedure

1. Open the Red Hat Customer Portal in your browser and log in to your Red Hat account.
2. Navigate to **Subscriptions**, which is in the upper-left corner of the Customer Portal page.
3. Click the **Subscription Allocations** tab.
4. From the **Name** column, select the name of the subscription allocation you want to modify.
5. Click the **Subscriptions** tab.
6. On the row of each subscription you want to remove, select the corresponding check box.
7. Click **Remove**, and then confirm removal.
8. If the modified subscription allocation has a corresponding subscription manifest on a Satellite Server, refresh the subscription manifest. For more information, see [Section 4.2.5, "Updating and Refreshing Subscription Manifests"](#).

### 4.1.4. Moving Subscriptions between Subscription Allocations

The following procedure explains how to move subscriptions from one subscription allocation to another.

#### Procedure

1. Open the Red Hat Customer Portal in your browser and log in to your Red Hat account.
2. Navigate to **Subscriptions**, which is in the upper-left corner of the Customer Portal page.
3. Click the **Subscription Allocations** tab.
4. From the **Name** column, select the name of the subscription allocation you want to modify.
5. Click the **Subscriptions** tab.
6. On the row of each subscription you want to move, select the corresponding check box.
7. Click **Move to Another Allocation**.
8. In the "Move entitlements from this subscription allocation to another" window, use the appropriate method to select the subscriptions you want to move:
  - If you want to select all subscriptions, select **Move All Entitlements**.

- If you want to specify quantities, enter the appropriate quantity on the row of each subscription under the **Quantity to Move** column.
  - If you want to remove a subscription from the selection, click the delete icon on the row of that subscription, and then click **OK** to confirm.
9. Click **Next: Select Destination Allocation**.
  10. From the table, select the subscription allocation you want to move the subscriptions to, and then click **Submit**.
  11. If the modified subscription allocation has a corresponding subscription manifest on a Satellite Server, refresh the subscription manifest. For more information, see [Section 4.2.5, “Updating and Refreshing Subscription Manifests”](#).

### 4.1.5. Using Future Dated subscriptions

You can use future-dated subscriptions in a subscription allocation. When you add future-dated subscriptions to content hosts before the expiry date of the existing subscriptions, you can have uninterrupted access to repositories.

Red Hat recommends that you manually attach the future-dated subscriptions to your content hosts before the current subscriptions expire and not to rely on the auto-attach method. For more information, see [Section 4.2.2, “Attaching Subscriptions to Content Hosts”](#).

### 4.1.6. Using Subscription Allocations to Manage Multiple Organizations

You can manage more than one organization if you have more than one subscription allocation. Satellite 6 requires a single allocation for each organization configured in your Satellite Server. The advantage of this is that each organization maintains completely separate subscriptions so that you can support multiple organizations, each with their own Red Hat accounts.

### 4.1.7. Exporting a Subscription Manifest from the Customer Portal

When viewing a subscription allocation that has at least one subscription, you can export the allocation as a subscription manifest.

A subscription manifest is a ZIP archive of the subscription allocation, containing encoded subscriptions, which can be imported into your Satellite Server.

#### Procedure

1. Open <https://access.redhat.com/> in your browser and log in to your Red Hat account.
2. Navigate to **Subscriptions**, which is in the upper-left corner of the Customer Portal page.
3. Click the **Subscription Allocations** tab.
4. From the **Name** column, select the name of the subscription allocation you want to export.
5. On the **Details** pane, click **Export Manifest** to download the subscription manifest file.

### 4.1.8. Importing a Subscription Manifest into Satellite Server

The following procedure explains how to import a Subscription Manifest into your Satellite Server.

## Prerequisite

You must have a subscription manifest file previously exported from the Customer Portal. For more information, see [Section 4.1.7, "Exporting a Subscription Manifest from the Customer Portal"](#) .

### For Web UI Users

1. In the Satellite web UI, ensure the context is set to the organization you want to use.
2. Navigate to **Content > Subscriptions**.
3. On the Red Hat Subscriptions page, click **Manage Manifest**.
4. In the Manage Manifest window, click **Browse**.
5. Navigate to the location containing the subscription manifest file, and then click **Open**.
6. If the Manage Manifest window does not close automatically, click **Close** to return to the Red Hat Subscriptions page.

### For CLI Users

The Red Hat Satellite 6 CLI requires the manifest to be on your Satellite Server.

1. Log in to your local client system, and then copy the subscription manifest file to your Satellite Server:

```
[user@client ~]$ scp ~/manifest_file.zip root@satellite.example.com:~/.
```

2. In a terminal, connect to your Satellite Server as the root user, and then import the subscription manifest file:

```
[root@satellite ~]# hammer subscription upload \  
--file ~/manifest_file.zip \  
--organization "organization_name"
```

After a few minutes, the CLI reports a successful manifest import.

You can now enable repositories and import Red Hat content. For more information, see [Chapter 5, Importing Red Hat Content](#) .

## 4.2. MANAGING SUBSCRIPTION ALLOCATIONS IN THE SATELLITE WEB UI

With Red Hat Satellite 6.4, you can now manage your subscription allocations directly in the Satellite web UI.

### 4.2.1. Searching for a Subscription in the Satellite Web UI

When you import a subscription manifest into your Satellite Server, the subscriptions from your manifest are listed on the Red Hat Subscriptions page. If you have a high volume of subscriptions, you can filter the results to find a specific subscription.

## Prerequisite

You must have a subscription manifest file previously imported to your Satellite Server. For more information, see [Section 4.1.8, “Importing a Subscription Manifest into Satellite Server”](#).

### Procedure

1. In the Satellite web UI, ensure the context is set to the organization you want to use.
2. Navigate to **Content > Subscriptions**.
3. On the Red Hat Subscriptions page, click the **Search** field to view the list search criteria for building your search query.
4. Select a search criteria to display further options.
5. When you have built your search query, click the search icon.

For example, if you place your cursor in the **Search** field, and select **expires**, then press the space bar, another list appears with the options of placing a **>**, **<**, or **=** characters. If you select, **>**, and press the space bar, another list of automatic options appear. You can also enter your own criteria.

## 4.2.2. Attaching Subscriptions to Content Hosts

Using activation keys is the recommended way to attach subscriptions to content hosts during provisioning. However, an activation key cannot update an existing host. If you need to attach new or additional subscriptions, such as future-dated subscriptions, to one host, follow the method below.

For more information about updating multiple hosts, see [Section 4.2.6, “Bulk Updating Content Hosts’ Subscriptions”](#).

For more information about activation keys, see [Chapter 9, Managing Activation Keys](#).

### Prerequisite

You must have a subscription manifest file previously imported to your Satellite Server. For more information, see [Section 4.1.8, “Importing a Subscription Manifest into Satellite Server”](#).

### Smart Management Subscriptions

In Satellite 6, the number of Red Hat Enterprise Linux Smart Management subscriptions correspond with how many systems Satellite can manage. You must maintain a Smart Management subscription for every Red Hat Enterprise Linux subscription that you want to manage with Satellite.

However, you are not required to attach Smart Management subscriptions to each content host. Smart Management subscriptions cannot attach automatically to content hosts in Satellite because they are not associated with any product certificates. Adding a Smart Management subscription to a content host does not provide any content or repository access. If you want, you can add a smart management subscription to a manifest for your own recording or tracking purposes.

### For Web UI Users

1. In the Satellite web UI, ensure the context is set to the organization you want to use.
2. Navigate to **Hosts > Content Hosts**.
3. On the row of each content host whose subscription you want to change, select the corresponding check box.

4. From the **Select Action** list, select **Manage Subscriptions**.
5. Optionally, enter a key and value in the **Search** field to filter the subscriptions displayed.
6. Select the check box to the left of the subscriptions to be added or removed and click **Add Selected** or **Remove Selected** as required.
7. Click **Done** to save the changes.

#### For CLI Users

1. In a terminal, connect to your Satellite Server as the root user, and then list the available subscriptions:

```
# hammer subscription list \  
--organization-id 1
```

2. Attach a subscription to the host:

```
# hammer host subscription attach \  
--host host_name \  
--subscription-id subscription_id
```

### 4.2.3. Adding Subscriptions to Subscription Allocations in the Satellite Web UI

The following procedure explains how to add subscriptions to a subscription allocation in the Satellite web UI.

#### Prerequisite

You must have a subscription manifest file imported to your Satellite Server. For more information, see [Section 4.1.8, "Importing a Subscription Manifest into Satellite Server"](#).

#### Procedure

1. In the Satellite web UI, ensure the context is set to the organization you want to use.
2. Navigate to **Content > Subscriptions**.
3. On the Red Hat Subscriptions page, click **Add Subscriptions**.
4. On the row of each subscription you want to add, enter the quantity under the **Quantity to Allocate** column.
5. Click **Submit**.

After a few minutes, the web UI reports that the action has been successful.

### 4.2.4. Removing Subscriptions from Subscription Allocations in the Satellite Web UI

The following procedure explains how to remove subscriptions from a subscription allocation in the Satellite web UI.





## NOTE

Manifests must not be deleted. If you delete the manifest from the Red Hat Customer Portal or in the Satellite web UI, all of the entitlements for all of your content hosts will be removed.

### Prerequisite

You must have a subscription manifest file imported to your Satellite Server. For more information, see [Section 4.1.8, "Importing a Subscription Manifest into Satellite Server"](#).

### Procedure

1. In the Satellite web UI, ensure the context is set to the organization you want to use.
2. Navigate to **Content** > **Subscriptions**.
3. On the row of each subscription you want to remove, select the corresponding check box.
4. Click **Delete**, and then confirm deletion.

After a few minutes, the web UI reports that the action has been successful.

### 4.2.5. Updating and Refreshing Subscription Manifests

Every time that you change a subscription allocation, you must refresh the manifest to reflect these changes. For example, you must refresh the manifest if you take any of the following actions:

- Renewing a subscription
- Adjusting subscription quantities
- Purchasing additional subscriptions

You can use refresh the manifest directly in the Satellite web UI. Alternatively, you can import an updated manifest that contains the changes. For more information about importing a manifest, see [Section 4.1.8, "Importing a Subscription Manifest into Satellite Server"](#).

### Procedure

1. In the Satellite web UI, ensure the context is set to the organization you want to use.
2. Navigate to **Content** > **Subscriptions**.
3. On the Red Hat Subscriptions page, click **Manage Manifest**.
4. In the Manage Manifest window, click **Refresh**.

After a few minutes, the web UI reports that the action has been successful.

### 4.2.6. Bulk Updating Content Hosts' Subscriptions

The methods described here are intended for post installation changes to multiple content hosts at the same time. You can use the web UI and the filter function to select the content hosts to be changed, or use the Hammer command-line tool's CSV file export function, edit the configuration settings in the CSV file, and upload the changes.

### For Web UI Users

1. In the Satellite web UI, ensure the context is set to the organization you want to use.
2. Navigate to **Hosts > Content Hosts**.
3. On the row of each content host whose subscription you want to change, select the corresponding check box.
4. From the **Select Action** list, select **Manage Subscriptions**.
5. Optionally, enter a key and value in the **Search** field to filter the subscriptions displayed.
6. Select the check box to the left of the subscriptions to be added or removed and click **Add Selected** or **Remove Selected** as required.
7. Click **Done** to save the changes.

### For CLI Users

1. Export the current state of content hosts to a CSV file.

```
# hammer --server https://satellite.example.com csv content-hosts --export --file content_hosts.csv
```

2. Change the required values in the CSV file. You can use an editor, with a CSV plug-in, or **sed** to change strings in the CSV file.

- a. Make a backup of the file:

```
# cp content_hosts.csv content_hosts.csv.backup
```

- b. Edit the string you want to change. For example:

```
# sed -i "s/1|RH1234|Red Hat Enterprise Linux Server/1|RH5678|Red Hat Enterprise Linux Server/g" content_hosts.csv
```

- c. Confirm only the required changes were made. For example:

```
# diff content_hosts.csv content_hosts.csv.backup
```

3. Upload the changed file to Satellite Server:

```
# hammer --server https://satellite.example.com csv content-hosts --file content_hosts.csv
```

## CHAPTER 5. IMPORTING RED HAT CONTENT

Our Satellite Server now has the necessary subscription information imported. We are ready to add content to our system. For this chapter, we explore the concept of a Definitive Media Library (DML) and how to synchronize content to create our DML.

### 5.1. CREATING A DEFINITIVE MEDIA LIBRARY

A DML is a repository that stores and protects the definitive, authorized versions of software and configurations. In other words, the DML acts as a master version of any content imported into Satellite. This includes Red Hat content, such as RPM files, kickstart trees, and ISO images. As mentioned in [Section 1.1, “Overview of Red Hat Satellite 6 Content Management”](#), Red Hat Satellite 6 stores and manages content in a DML.

### 5.2. USING PRODUCTS AND REPOSITORIES IN SATELLITE

In Satellite, we use the concept of a **Product** as an organizational unit to group multiple repositories together. Such repository collections are analogous to the concept of real life products. For example, if we view Red Hat Enterprise Linux Server as a Product in Satellite, the repositories for that product might consist of different versions (6.0, 6.1, 7.0), different architectures (i386, x86\_64, s390x, arm), and different add-ons (Optional repositories, Supplementary repositories, Virt V2V tools). This unifies all related repositories within the DML. Using Products ensures repositories that depend on each other are synchronized together. For Red Hat repositories, products are created automatically after enabling the repository.

In this chapter, our aim is to start creating our DML using Red Hat content. To do this, we synchronize our DML with Red Hat’s Products and their repositories.

### 5.3. SYNCHRONIZING CONTENT

When you select which repositories form the DML, Satellite Server synchronizes its own repositories with the repositories on the Red Hat CDN. This ensures that Satellite Server retains an exact copy of Red Hat’s repositories as a part of its DML. Satellite Server fetches this repository information and stores it on Satellite Server’s file system. After an initial synchronization, you can create a synchronization plan that checks to ensure the repositories in the DML is up to date with the CDN’s repositories.

It is possible to perform an initial synchronization using ISO images. See [Appendix C, \*Importing Content ISOs into a Connected Satellite\*](#) for more information on using Content ISOs. For locations with bandwidth limitations, using an **On Demand** or **Background** download policy as described below might be quicker than downloading and importing Content ISOs.

#### 5.3.1. Disabling the Global HTTP Proxy

If you want to synchronize local repositories without using the configured proxy, you can ignore the global proxy that is set when you run the **satellite-installer** tool.

#### Procedure

To ignore the global HTTP proxy, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Products** and select the product that you want to change.

2. Click the **Repositories** tab, and select the repository that you want to change.
3. In the **Sync Settings** area, navigate to **Ignore Global HTTP Capsule**. Click the edit icon, select the check box, and then click **Save**.

### For CLI users

```
# hammer repository update --id Repository_ID --ignore-global-proxy yes
```

## 5.4. CONFIGURING DOWNLOAD POLICIES

Red Hat Satellite provides multiple download policies for synchronizing RPM content. For example, you might aim to save time and only download the content metadata while deferring the actual content download for later.

Satellite Server offers the following policies:

- **Immediate** – Satellite Server downloads all metadata and packages during synchronization.
- **On Demand** – Satellite Server only downloads the metadata during synchronization. Satellite Server only fetches and stores packages on the file system when Capsules or directly connected clients request them. This setting has no effect if you set a corresponding repository on a Capsule to **Immediate** because Satellite Server is forced to download all the packages.
- **Background** – Satellite Server creates a background task to download all packages after the initial synchronization.

The latter two policies act as a *Lazy Synchronization* feature because they save time synchronizing content. The lazy sync feature must only be used for **yum** repositories. The packages can be added to Content Views and promoted to life cycle environments as normal.

Capsule Server offers the following policies:

- **Immediate** – Capsule Server downloads all metadata and packages during synchronization. Do not use this setting if the corresponding repository on Satellite Server is set to **On Demand** as Satellite Server is forced to download all the packages.
- **On Demand** – Capsule Server only downloads the metadata during synchronization. The Capsule only fetches and stores packages on the file system when directly connected clients request them.



### NOTE

When using an **On Demand** download policy, content is downloaded from Satellite Server if it is not available on the Capsule Server.

- **Background** – Capsule Server creates a background task to download all packages after the initial synchronization.
- **Inherit** – Capsule Server inherits the download policy for the repository from the corresponding repository on Satellite Server.

These policies are not available if a Capsule was installed or updated with **--enable-foreman-proxy-plugin-pulp** set to false.

### 5.4.1. Changing the Default Download Policy

You can set the default download policy globally so that Satellite applies it to new repositories you create in all organizations. Changing the default value does not change existing settings.

#### For Web UI Users

1. In the Satellite web UI, navigate to **Administer** > **Settings**.
2. Click the **Content** tab, locate **default\_download\_policy**.
3. In the **Value** field, click the edit icon.
4. Select the required download policy, and then click **Save**.

#### For CLI Users

To change the default download policy to one of **immediate**, **on\_demand**, or **background**, enter the following command:

```
# hammer settings set \
--name default_download_policy \
--value immediate
```

### 5.4.2. Changing the Download Policy for a Repository

Use this procedure to change the download policy for a specific repository.

#### For Web UI Users

1. In the web UI, navigate to **Content** > **Products**, and click the required product name.
2. Click the required repository name, locate the **Download Policy** field, and click the edit icon.
3. From the list, select the required download policy and then click **Save**.

#### For CLI Users

1. List the repositories for an organization:

```
# hammer repository list \
--organization-label organization-label
```

2. Change the download policy for a repository to one of **immediate**, **on\_demand**, or **background**:

```
# hammer repository update \
--organization-label organization-label \
--product "Red Hat Enterprise Linux Server" \
--name "Red Hat Enterprise Linux 7 Server Kickstart x86_64 7.5" \
--download-policy immediate
```

## 5.5. SELECTING RED HAT REPOSITORIES TO SYNCHRONIZE

To select the repositories to synchronize you must first identify the product that contains the repository, and then enable that repository based on the relevant release version and base architecture.



## IMPORTANT

If using a Disconnected Satellite, you need to import the Content ISOs for Red Hat Satellite and change the CDN URL on Satellite Server before synchronizing content. For more information, see [Appendix B, Importing Content ISOs into a Disconnected Satellite](#).

### For Web UI Users

1. Navigate to **Content > Red Hat Repositories**.

The Red Hat Repositories page displays the following two panes: the Available Repositories and the Enabled Repositories panes. The Available Repositories pane contains a list of all subscribed products that provide RPM content. The Enabled Repositories pane contains a list of enabled repositories.

The relationship between products and specific repositories is connected through a cascading hierarchy that is displayed in the Available Repositories pane.

2. To find the **Red Hat Enterprise Linux 7 Server (RPMs)** repositories, use one of the following methods:
  - In the **Search** field, enter the following repository name: **rhel-7-server-rpms**.
  - Toggle the **Recommended Repositories** button to the ON position to view a list of repositories that are used in a typical setup.
3. In the Available Repositories pane, click on the **Red Hat Enterprise Linux 7 Server (RPMs)** to expand the repository set.
4. For the **x86\_64 7Server** entry, click the **Enable** icon to enable the repository. This enables the latest RPM files for Red Hat Enterprise Linux 7.



## NOTE

The difference between associating Red Hat Enterprise Linux Operating System with either 7 Server repositories or 7.X repositories is that 7 Server repositories contain all the latest updates while Red Hat Enterprise Linux 7.X repositories stop getting updates after the next minor version release. Note that Kickstart repositories only have minor versions.

### For CLI Users

The relationship between products and repositories is the same. To search for your product, enter the following command:

```
# hammer product list --organization "ACME"
```

List the repository set for the product:

```
# hammer repository-set list \  
--product "Red Hat Enterprise Linux Server" \  
--organization "ACME"
```

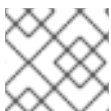
This displays the repositories in the product's repository set, including their name and ID number. Enable the repository using either the name or ID number. Also include the release version (**7Server**) and base architecture (**x86\_64**). For example:

```
# hammer repository-set enable \
--name "Red Hat Enterprise Linux 7 Server (RPMs)" \
--releasever "7Server" \
--basearch "x86_64" \
--product "Red Hat Enterprise Linux Server" \
--organization "ACME"
```

For this scenario, use either the Web UI or the CLI to enable the following repositories for ACME:

Repository	Type	Description
Red Hat Enterprise Linux 7 Server RPMs x86_64 7Server	RPM	Repositories for the latest version of Red Hat Enterprise Linux 7. Use the <b>7 Server</b> repository instead of the <b>7.2</b> repository so that you receive continuous package updates.
Red Hat Satellite Tools 6.4 for RHEL 7 Server RPMs x86_64	RPM	The Satellite Tools repository, which contains system management agents and tools for client systems. After provisioning a new system, Satellite installs tools such as <b>katello-agent</b> and Puppet to the client. Use the <b>7 Server</b> repository instead of the <b>7.2</b> repository so that you receive continuous package updates.
Red Hat Enterprise Linux 7.2 Kickstart x86_64 7Server	Kickstart	The kickstart tree for Red Hat Enterprise Linux 7.2. Use this as installation media for provisioning new systems over PXE.

These repositories provide some initial content for our scenario's DML. You can select more repositories based upon your needs.



#### NOTE

For this scenario, all repositories use **x86\_64** as the base architecture.

## 5.6. SYNCHRONIZING RED HAT REPOSITORIES

We have enabled specific repositories to form our initial DML. Now we synchronize the repositories with the Red Hat CDN's repositories.

### For Web UI Users

Navigate to **Content > Products** and select **Red Hat Enterprise Linux Server**. This displays all enabled

repositories in our product. Select all repositories and click **Sync Now**. You can also view the progress of the synchronization in the Web UI. Navigate to **Content > Sync Status** and expanding the Product/Repository tree (or click **Expand All**).

### For CLI Users

Synchronize the enabled repositories in the Red Hat Enterprise Linux Server product:

```
# hammer product synchronize \
--name "Red Hat Enterprise Linux Server" \
--organization "ACME"
```

You can also synchronize each repository individually. List all repositories in the product, then synchronize using the ID number for the corresponding repositories. For example:

```
# hammer repository list \
--product "Red Hat Enterprise Linux Server" \
--organization "ACME"
# hammer repository synchronize \
--name "Red Hat Enterprise Linux 7 Server RPMs x86_64 7Server" \
--product "Red Hat Enterprise Linux Server" \
--organization "ACME"
```

The synchronization duration depends on the size of each repository and the speed of your network connection. The following table provides estimates of how long it would take to synchronize content, depending on the available Internet bandwidth:

	Single Package (10Mb)	Minor Release (750Mb)	Major Release (6Gb)
256 Kbps	5 Mins 27 Secs	6 Hrs 49 Mins 36 Secs	2 Days 7 Hrs 55 Mins
512 Kbps	2 Mins 43.84 Secs	3 Hrs 24 Mins 48 Secs	1 Day 3 Hrs 57 Mins
T1 (1.5 Mbps)	54.33 Secs	1 Hr 7 Mins 54.78 Secs	9 Hrs 16 Mins 20.57 Secs
10 Mbps	8.39 Secs	10 Mins 29.15 Secs	1 Hr 25 Mins 53.96 Secs
100 Mbps	0.84 Secs	1 Min 2.91 Secs	8 Mins 35.4 Secs
1000 Mbps	0.08 Secs	6.29 Secs	51.54 Secs

A manual synchronization is often required for the initial content import into the DML. However, it is recommended to create a synchronization plan to ensure that our DML updates on a regular basis.



#### NOTE

You can change the download policy for a Red Hat repository. Select a repository in the **Red Hat Enterprise Linux Server** product and navigate to the **Download Policy** field. If using the CLI, enter the **hammer repository update** command with the **--download-policy** option.



### 5.6.1. Recovering a Repository

In the case of repository corruption, you can recover it by using an advanced synchronization, which has three options:

- **Optimized Sync** - it synchronizes the repository bypassing RPMs that have no detected differences from the upstream RPMs.
- **Complete Sync** - it synchronizes all RPMs regardless of detected changes. Use this option if specific RPMs could not be downloaded to the local repository even though they exist in the upstream repository.
- **Validate Content Sync** - it synchronizes all RPMs and then verifies the checksum of all RPMs locally. If the checksum of an RPM differs from the upstream, it re-downloads the RPM. This option is relevant for **yum** repositories only. Use this option if you have one of the following errors:
  - Specific RPMs cause a **404** error while synchronizing with **yum**.
  - **Package does not match intended download** error, which means that specific RPMs are corrupted.

To synchronize a specific repository with an advanced option:

#### For Web UI users

1. Navigate to **Content > Products**.
2. Select the product containing the corrupted repository.
3. Click on the name of a repository you want to synchronize.
4. Extend the **Select Action** menu and select **Advanced Sync**.
5. Select the option and click **Sync**.

#### For CLI users

1. Obtain a list of repository IDs:

```
# hammer repository list --organization "Default Organization"
```

2. Synchronize a corrupted repository using the necessary option:

- For the optimized synchronization:

```
# hammer repository synchronize --incremental true --id 1
```

- For the complete synchronization:

```
# hammer repository synchronize --skip-metadata-check true --id 1
```

- For the validate content synchronization:

```
# hammer repository synchronize --validate-contents true --id 1
```

## 5.6.2. Limiting Synchronization Speed

You can control the speed of synchronization to avoid exhaustion of available bandwidth and prevent other performance issues. This is done by configuring **PULP\_CONCURRENCY** and **max\_speed** parameters. Note that these settings are overwritten on an upgrade. It is recommended to backup changed files prior to an upgrade to be able to restore the configuration.

1. To control the number of synchronization jobs that run in parallel, configure the **PULP\_CONCURRENCY** parameter in the `/etc/default/pulp_workers` file. For example, to set the number of jobs that run in parallel to 1, change **PULP\_CONCURRENCY** to 1:

```
PULP_CONCURRENCY=1
```

By default, on a system with less than 8 CPUs, **PULP\_CONCURRENCY** is set to the number of CPUs. On a system with more than 8 CPUs, it is set to 8.

2. To set the maximum network speed for synchronizing in bytes per second, configure the **max\_speed** parameter. This parameter must be configured separately for each importer in the `/etc/pulp/server/plugins.conf.d/` directory.
  - a. For example, to set the maximum speed for synchronizing RPM content to 10 bytes per second, set the **"max\_speed"** parameter in the `/etc/pulp/server/plugins.conf.d/yum_importer.json` file to 10:

```
# cat /etc/pulp/server/plugins.conf.d/yum_importer.json
{
  "proxy_host": null,
  "proxy_port": null,
  "proxy_username": null,
  "proxy_password": null,
  "max_speed": 10
}
```

- b. Verify the syntax of the file after editing:

```
# json_verify < /etc/pulp/server/plugins.conf.d/yum_importer.json
JSON is valid
```

3. Restart Satellite services to apply the changes:

```
# foreman-maintain service restart
```

## 5.7. CREATING A SYNCHRONIZATION PLAN

A synchronization plan checks and updates the content in your DML at a regularly scheduled date and time. Red Hat Satellite 6 provides users with the ability to create a synchronization plan and assign products to it.

### For Web UI Users

Navigate to **Content > Sync Plans** and click **New Sync Plan**. The UI provides a set of fields where you can input details about your synchronization plan:

- **Name** - A plain text name for the plan. Enter **Example Plan**.

- **Description** - A plain text description of the plan. Enter **Example Plan for ACME's repositories**.
- **Interval** - Defines when to run the synchronization. Select **daily**.
- **Start Date** and **Start Time** - Defines when to run the synchronization. We already completed a synchronization today, so set the synchronization for tomorrow at 1:00 (1AM).

Click **Save** to create a plan. The plan details page displays along with two tabs for **Details** and **Products**.

Now add your products. Click the **Products** tab, then click **Add**. Select the **Red Hat Enterprise Linux Server** product and click **Add Selected**.

### For CLI Users

To create the synchronization plan, enter the following command:

```
# hammer sync-plan create \  
--name "Red Hat Products 2" \  
--description "Example Plan for ACME's Red Hat Products" \  
--interval daily \  
--sync-date "2016-02-01 01:00:00" \  
--enabled true \  
--organization "ACME"
```

Then assign the Red Hat Enterprise Linux Server product to it:

```
# hammer product set-sync-plan \  
--name "Red Hat Enterprise Linux Server" \  
--sync-plan "Red Hat Products" \  
--organization "ACME"
```

Now Satellite Server checks its DML content against the Red Hat CDN on a daily basis and keeps its Red Hat repositories up to date.

## CHAPTER 6. IMPORTING CUSTOM CONTENT

This chapter outlines how you can import custom content of different types to Satellite. If you want to import ISOs, Puppet modules, or different content types to Satellite, it is done with largely the same procedures in this chapter.

For example, you can use the following chapters for information on specific types of custom content but the underlying procedures are the same:

- [Chapter 11, \*Managing OSTree Content\*](#)
- [Chapter 12, \*Managing ISO Images\*](#)
- [Chapter 13, \*Managing Custom File Type Content\*](#)
- [Chapter 14, \*Managing Custom Puppet Content\*](#)

### 6.1. USING CUSTOM PRODUCTS IN SATELLITE

Both Red Hat content and custom content in Red Hat Satellite 6 have similarities:

- The relationship between a product and its repositories is the same and the repositories still require synchronization.
- Custom products require a subscription for clients to access, similar to subscriptions to Red Hat Products. Red Hat Satellite 6 creates a subscription for each custom product you create.

For more information about creating and packaging RPMs, see the [RPM Packaging Guide](#) in the Red Hat Enterprise Linux documentation.

### 6.2. IMPORTING CUSTOM SSL CERTIFICATES

Before you create custom content, ensure that you import any Custom SSL certificates that you require.

If you require SSL certificates and keys to download RPMs, you can add them to Satellite.

1. In the Satellite web UI, navigate to **Content > Content Credentials**. In the Content Credentials window, click **Create Content Credential**.
2. In the **Name** field, enter a name for your SSL certificate.
3. From the **Type** list, select **SSL Certificate**.
4. In the **Content Credentials Content** field, paste your SSL certificate, or click **Browse** to upload your SSL certificate.
5. Click **Save**.

### 6.3. IMPORTING A CUSTOM GPG KEY

Before you create custom content, ensure that you import any GPG keys that you require.

#### Prerequisites

1. Download a copy of the version specific repository package to your client system:

```
$ wget http://www.example.com/9.5/example-9.5-2.noarch.rpm
```

2. Extract the RPM file without installing it:

```
$ rpm2cpio example-9.5-2.noarch.rpm | cpio -idmv
```

The GPG key is located relative to the extraction at **etc/pki/rpm-gpg/RPM-GPG-KEY-EXAMPLE-95**.

## Procedure

To import a GPG key, complete the following procedure:

1. In the Satellite web UI, navigate to **Content > Content Credentials** and in the upper-right of the window, click **Create Content Credential**.
2. Enter the name of your repository and select **GPG Key** from the **Type** list.
3. Either paste the GPG key into the **Content Credential Contents** field, or click **Browse** and select the GPG key file that you want to import.

If your custom repository contains content signed by multiple GPG keys, you must enter all required GPG keys in the **Content Credential Contents** field with new lines between each key, for example:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFy/HE4BEADttv2TCPzVrre+aJ9f5QsR6oWZMm7N5Lwxjm5x5zA9BLiPPGFN
4aTUR/g+K1S0aqCU+ZS3Rnxb+6fnBxD+COH9kMqXHi3M5UNzbp5WhCdUpISXjipU
XIFFWBPuBfyr/FKRknFH15P+9kLZLxCpVZZLsweLWCuw+JKCMmnA
=F6VG
-----END PGP PUBLIC KEY BLOCK-----

-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFw467UBEACmREzDeK/kuScCmfJfHJa0Wgh/2fbJLLt3KSvsgDhORlptf+PP
OTFDIKuLkXj99ZYG5xMnBG47C7ByoMec1j94YeXczuBbynOyyPlvduma/zf8oB9e
WI5GnzcLGAUSRamfqGUWcyMMinHHIKlc1X1P4I=
=WPpl
-----END PGP PUBLIC KEY BLOCK-----
```

4. Click **Save**.

## For CLI Users

1. Copy the GPG key to your Satellite Server:

```
$ scp ~/etc/pki/rpm-gpg/RPM-GPG-KEY-EXAMPLE-95 root@satellite.example.com:~/.
```

2. Upload the GPG key to Satellite:

```
# hammer gpg create \
--key ~/RPM-GPG-KEY-EXAMPLE-95 \
--name "My_Repository" \
```

```
--organization "My_Organization"
```

## 6.4. CREATING A CUSTOM PRODUCT

Use this procedure to create a custom product that you can then add repositories to.

### Procedure

To create a custom product, complete the following procedure:

1. In the Satellite web UI, navigate to **Content > Products**, click **New Product**.
2. In the **Name** field, enter a name for the product. Red Hat Satellite 6 automatically completes the **Label** field based on what you have entered for **Name**.
3. Optional: From the **GPG Key** list, select the GPG key for the product.
4. Optional: From the **SSL CA Cert** list, select the SSL CA certificate for the product.
5. Optional: From the **SSL Client Cert** list, select the SSL client certificate for the product.
6. Optional: From the **SSL Client Key** list, select the SSL client key for the product.
7. Optional: From the **Sync Plan** list, select an existing sync plan or click **Create Sync Plan** and create a sync plan for your product requirements.
8. In the **Description** field, enter a description of the product.
9. Click **Save**.

### For CLI Users

To create the product, enter the following command:

```
# hammer product create \  
--name "My_Product" \  
--sync-plan "Example Plan" \  
--description "Content from My Repositories" \  
--organization "My_Organization"
```

## 6.5. ADDING A CUSTOM RPM REPOSITORY

Use this procedure to add a custom RPM repository in Satellite.

The Products window in the Satellite web UI also provides a **Repo Discovery** function that finds all repositories from a URL and you can select which ones to add to your custom product. For example, you can use the **Repo Discovery** to search, for example, <http://yum.postgresql.org/9.5/redhat/> and list all repositories for different Red Hat Enterprise Linux versions and architectures. This helps users save time importing multiple repositories from a single source.

### Support for Custom RPMs

Red Hat does not support the upstream RPMs directly from third-party sites. These RPMs are used to demonstrate the synchronization process. For any issues with these RPMs, contact the third-party developers.

## Procedure

1. In the Satellite web UI, navigate to **Content** > **Products** and select the product that you want to use, and then click **Create Repository**.
2. In the **Name** field, enter a name for the repository. Red Hat Satellite 6 automatically completes the **Label** field based on what you have entered for **Name**.
3. From the **Type** list, select the type of repository. You can select either a repository for RPM files (**yum**), Puppet modules (**puppet**), or Docker images (**docker**).
4. In the **URL** field, enter the URL of the external repository to use as a source.
5. From the **Download Policy** list, select the type of synchronization Satellite Server performs.
6. Ensure that the **Mirror on Sync** check box is selected. This ensures that the content that is no longer part of the upstream repository is removed during synchronization.
7. From the **Checksum** list, select the checksum type for the repository.
8. Optional: If you want, you can clear the **Publish via HTTP** check box to disable this repository from publishing through HTTP.
9. Optional: From the **GPG Key** list, select the GPG key for the product.
10. Click **Save**.

If you want to perform an immediate synchronization, in your product window, click **Sync Now**.

## For CLI Users

1. Enter the following command to create the repository:

```
# hammer repository create \
  --name "My_Repository" \
  --content-type "yum" \
  --publish-via-http true \
  --url http://yum.postgresql.org/9.5/redhat/rhel-7-x86_64/\
  --gpg-key "My_Repository" \
  --product "My_Product" \
  --organization "My_Organization"
```

2. Synchronize the repository:

```
# hammer repository synchronize \
  --name "My_Repository" \
  --product "My_Product" \
  --organization "My_Organization"
```

## CHAPTER 7. MANAGING APPLICATION LIFE CYCLES

This chapter outlines the application life cycle in Satellite and how to create and remove application life cycles for Satellite and Capsule.

### 7.1. APPLICATION LIFE CYCLE OVERVIEW

The *application life cycle* is a concept central to Red Hat Satellite 6's content management functions. The application life cycle defines how a particular system and its software look at a particular stage. For example, an application life cycle might be simple; you might only have a development stage and production stage. In this case the application life cycle might look like this:

- Development
- Production

However, a more complex application life cycle might have further stages, such as a phase for testing or a beta release. This adds extra stages to the application life cycle:

- Development
- Testing
- Beta Release
- Production

Red Hat Satellite 6 provides methods to customize each application life cycle stage so that it suits your specifications.

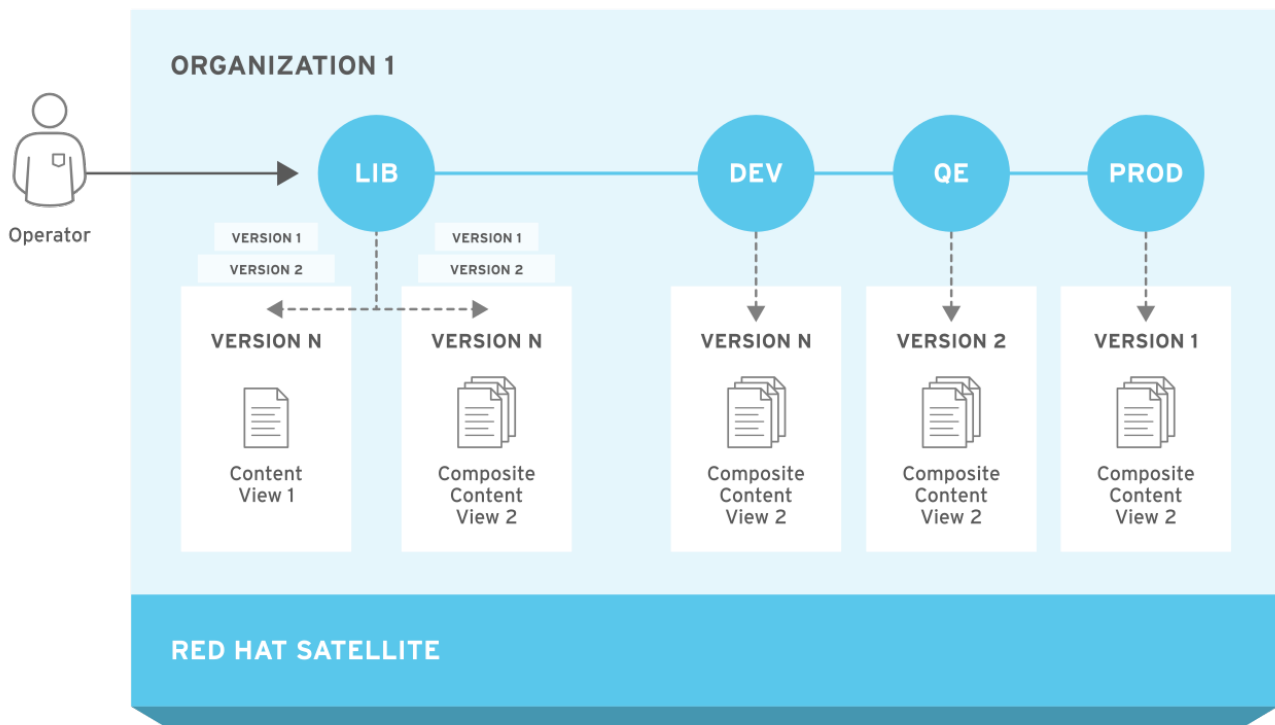
Each stage in the application life cycle is called an *environment* in Red Hat Satellite 6. Each environment uses a specific collection of content. Red Hat Satellite 6 defines these content collections as a Content View. Each Content View acts as a filter where you can define what repositories, packages, and Puppet modules to include in a particular environment. This provides a method for you to define specific sets of content to designate to each environment.

For example, an email server might only require a simple application life cycle where you have a production-level server for real-world use and a test server for trying out the latest mail server packages. When the test server passes the initial phase, you can set the production-level server to use the new packages.

Another example is a development life cycle for a software product. To develop a new piece of software in a development environment, test it in a quality assurance environment, pre-release as a beta, then release the software as a production-level application.



Figure 7.1. The Red Hat Satellite 6 Application Life Cycle



## 7.2. PROMOTING CONTENT ACROSS THE APPLICATION LIFE CYCLE

In the application life cycle chain, when content moves from one environment to the next, this is called *promotion*.

### Example Content Promotion Across Satellite Life Cycle Environments

Each environment contains a set of systems registered to Red Hat Satellite 6. These systems only have access to repositories relevant to their environment. When you promote packages from one environment to the next, the target environment's repositories receive new package versions. As a result, each system in the target environment can update to the new package versions.

Development	Testing	Production
<code>example_software-1.1-0.noarch.rpm</code>	<code>example_software-1.0-0.noarch.rpm</code>	<code>example_software-1.0-0.noarch.rpm</code>

After completing development on the patch, you promote the RPM to the Testing environment so the Quality Engineering team can review the patch. The application life cycle then contains the following package versions in each environment:

Development	Testing	Production
<code>example_software-1.1-0.noarch.rpm</code>	<code>example_software-1.1-0.noarch.rpm</code>	<code>example_software-1.0-0.noarch.rpm</code>

While the Quality Engineering team reviews the patch, the Development team starts work on `example_software 2.0`. This results in the following application life cycle:

Development	Testing	Production
<b>exampleware-2.0-0.noarch.rpm</b>	exampleware-1.1-0.noarch.rpm	exampleware-1.0-0.noarch.rpm

The Quality Engineering team completes their review of the patch. Now *example\_software* 1.1 is ready to release. You promote 1.1 to the *Production* environment:

Development	Testing	Production
example_software-2.0-0.noarch.rpm	example_software-1.1-0.noarch.rpm	<b>example_software-1.1-0.noarch.rpm</b>

The Development team completes their work on *example\_software* 2.0 and promotes it to the Testing environment:

Development	Testing	Production
example_software-2.0-0.noarch.rpm	<b>example_software-2.0-0.noarch.rpm</b>	example_software-1.1-0.noarch.rpm

Finally, the Quality Engineering team reviews the package. After a successful review, promote the package to the *Production* environment:

Development	Testing	Production
exampleware-2.0-0.noarch.rpm	exampleware-2.0-0.noarch.rpm	<b>exampleware-2.0-0.noarch.rpm</b>

For more information, see [Section 8.3, “Promoting a Content View”](#).

## 7.3. CREATING A LIFE CYCLE ENVIRONMENT PATH

To create an application life cycle for developing and releasing software, use the *Library* environment as the initial environment to create environment paths. Then optionally add additional environments to the environment paths.

### Procedure

1. In the Satellite web UI, navigate to **Content > Lifecycle Environments**.
2. Click **New Environment Path** to start a new application life cycle.
3. In the **Name** field, enter a name for your environment.
4. In the **Description** field, enter a description for your environment.
5. Click **Save**.

- Optional: To add an environment to the environment path, click **Add New Environment**, complete the **Name** and **Description** fields, and select the prior environment from the **Prior Environment** list.

#### For CLI Users

- To create an environment path, enter the **hammer lifecycle-environment create** command and specify the Library environment with the **--prior** option:

```
# hammer lifecycle-environment create \  
--name "Environment Path Name" \  
--description "Environment Path Description" \  
--prior "Library" \  
--organization "My_Organization"
```

- Optional: To add an environment to the environment path, enter the **hammer lifecycle-environment create** command and specify the parent environment with the **--prior** option:

```
# hammer lifecycle-environment create \  
--name "Environment Name" \  
--description "Environment Description" \  
--prior "Prior Environment Name" \  
--organization "My_Organization"
```

- To view the chain of the life cycle environment, enter the following command:

```
# hammer lifecycle-environment paths --organization "My_Organization"
```

## 7.4. REMOVING LIFE CYCLE ENVIRONMENTS FROM SATELLITE SERVER

Use this procedure to remove a life cycle environment. You can delete only the latest environment in an environment path. For example, if three environments exist in the order **Library**, **Dev**, and **Prod**, you must delete **Prod** before you can delete **Dev**. You cannot delete the **Library** environment.

### Procedure

To remove a life cycle environment, complete the following steps:

- In the Satellite web UI, navigate to **Content > Life Cycle Environments**.
- Click the name of the life cycle environment that you want to remove, and then click **Remove Environment**.
- Click **Remove** to remove the environment.

#### For CLI Users

- List the life cycle environments for your organization and note the name of the life cycle environment you want to remove:

```
# hammer lifecycle-environment list --organization "My_Organization"
```

- Use the **hammer lifecycle-environment delete** command to remove an environment:

```
# hammer lifecycle-environment delete \
--name "your_environment" \
--organization "My_Organization"
```

## 7.5. REMOVING LIFE CYCLE ENVIRONMENTS FROM CAPSULE SERVER

When life cycle environments are no longer relevant to the host system or environments are added incorrectly to Capsule Server, you can remove the life cycle environments from Capsule Server.

You can use both the Satellite web UI and the Hammer to remove life cycle environments from Capsule.

### Procedure

To remove a life cycle environment from Capsule Server, complete the following step:

1. In the Satellite web UI, navigate to **Infrastructure** > **Capsules**, and select the Capsule that you want to remove a life cycle from.
2. Click **Edit** and click the **Life Cycle Environments** tab.
3. From the right menu, select the life cycle environments that you want to remove from Capsule, and then click **Submit**.
4. To synchronize Capsule's content, click the **Overview** tab, and then click **Synchronize**.
5. Select either **Optimized Sync** or **Complete Sync**.

### For CLI Users

To remove a life cycle environment from Capsule Server, complete the following steps:

1. Select the Capsule Server that you want from the list and take note of its **id**:

```
# hammer capsule list
```

2. To verify the Capsule Server's details, enter the following command:

```
# hammer capsule info --id capsule_id
```

3. Verify the list of life cycle environments currently attached to the Capsule Server and take note of the **environment id**:

```
# hammer capsule content lifecycle-environments \
--id capsule_id
```

4. Remove the life cycle environment from Capsule Server:

```
# hammer capsule content remove-lifecycle-environment \
--id capsule_id \
--environment-id environment_id
```

Repeat this step for every life cycle environment that you want to remove from Capsule Server.

5. Synchronize the content from Satellite Server's environment to Capsule Server:

```
# hammer capsule content synchronize \  
--id capsule_id
```

## 7.6. ADDING LIFE CYCLE ENVIRONMENTS TO CAPSULE SERVERS

If your Capsule Server has the content functionality enabled, you must add an environment so that Capsule can synchronize content from Satellite Server and provide content to host systems.

Do not assign the *Library* lifecycle environment to your Capsule Server because it triggers an automated Capsule sync every time the CDN updates a repository. This might consume multiple system resources on Capsules, network bandwidth between Satellite and Capsules, and available disk space on Capsules.

You can use Hammer CLI on Satellite Server or the Satellite web UI.

### Procedure

To add a life cycle environment to Capsule Server, complete the following step:

1. In the Satellite web UI, navigate to **Infrastructure** > **Capsules**, and select the Capsule that you want to add a life cycle to.
2. Click **Edit** and click the **Life Cycle Environments** tab.
3. From the left menu, select the life cycle environments that you want to add to Capsule, and then click **Submit**.
4. To synchronize Capsule's content, click the **Overview** tab, and then click **Synchronize**.
5. Select either **Optimized Sync** or **Complete Sync**.

### For CLI Users

1. To display a list of all Capsule Servers, enter the following command:

```
# hammer capsule list
```

Note the ID that returns.

2. Using the ID, verify the details of your Capsule Server:

```
# hammer capsule info --id capsule_id
```

3. Verify the life cycle environments available and note the environment ID:

```
# hammer capsule content available-lifecycle-environments \  
--id capsule_id
```

4. To view the life cycle environments available for your Capsule Server, enter the following command and note the ID and the organization name:

```
# hammer capsule content available-lifecycle-environments --id capsule_id
```

5. Add the life cycle environment to your Capsule Server:

```
# hammer capsule content add-lifecycle-environment \  
--id capsule_id --organization "My_Organization" \  
--environment-id environment_id
```

Repeat for each life cycle environment you want to add to Capsule Server.

To synchronize all content from your Satellite Server environment to Capsule Server, enter the following command:

```
# hammer capsule content synchronize --id capsule_id
```

To synchronize a specific life cycle environment from your Satellite Server to Capsule Server, enter the following command:

```
# hammer capsule content synchronize --id external_capsule_id \  
--environment-id environment_id
```

## CHAPTER 8. MANAGING CONTENT VIEWS

Red Hat Satellite 6 uses Content Views to create customized repositories from the core repositories in your Definitive Media Library (DML). To do this, you must define which repositories to use and then apply certain filters to the content. These filters include both package filters, package group filters, and errata filters. You can use Content Views to define which software versions a particular environment uses. For example, a *Production* environment might use a Content View containing older package versions, while a *Development* environment might use a Content View containing newer package versions.

Each Content View creates a set of repositories across each environment, which Satellite Server stores and manages. When you promote a Content View from one environment to the next environment in the application life cycle, the respective repository on Satellite Server updates and publishes the packages.

	Development	Testing	Production
Content View Version and Contents	Version 2 - <i>example_software-1.1-0.noarch.rpm</i>	Version 1 - <i>example_software-1.0-0.noarch.rpm</i>	Version 1 - <i>example_software-1.0-0.noarch.rpm</i>

The repositories for Testing and Production contain the ***example\_software-1.0-0.noarch.rpm*** package. If you promote Version 2 of the Content View from Development to Testing, the repository for Testing regenerates and then contains the ***example\_software-1.1-0.noarch.rpm*** package:

	Development	Testing	Production
Content View Version and Contents	Version 2 - <i>example_software-1.1-0.noarch.rpm</i>	<b>Version 2 - <i>example_software-1.1-0.noarch.rpm</i></b>	Version 1 - <i>example_software-1.0-0.noarch.rpm</i>

This ensures systems are designated to a specific environment but receive updates when that environment uses a new version of the Content View.

The general workflow for creating Content Views for filtering and creating snapshots is as follows:

1. Create a Content View.
2. Add the repository and the Puppet modules that you want to the Content View.
3. Optionally, create one or more filters to refine the content of the Content View.
4. Publish the Content View.
5. Optionally, promote the Content View to another environment.
6. Attach the content host to the Content View.

If a repository is not associated with the Content View, the file **`/etc/yum.repos.d/redhat.repo`** remains empty and systems registered to it cannot receive updates.

Hosts can only be associated with a single Content View. To associate a host with multiple Content Views, create a composite Content View. For more information, see [Section 8.5, "Creating a Composite Content View"](#).

## 8.1. CREATING A CONTENT VIEW

Use this procedure to create a simple Content View.

### Procedure

To create a content view complete the following steps:

1. In the Satellite web UI, navigate to **Content** > **Content Views** and click **Create New View**.
2. In the **Name** field, enter a name for the view. Red Hat Satellite 6 automatically completes the **Label** field from the name you enter.
3. In the **Description** field, enter a description of the view.
4. Click **Save** to create the Content View.
5. In the **Repository Selection** area, select the repositories that you want to add to your Content View, then click **Add Repositories**.
6. Click **Publish New Version** and in the **Description** field enter information about the version to log changes.
7. Optional: to force metadata regeneration on Yum repositories, select the **Force Yum Metadata Regeneration** check box.
8. Click **Save**.

You can view the Content View in the Content Views window. To view more information about the Content View, click the Content View name.

To register a host to your content view, see [Registering a Host](#) in the *Managing Hosts* guide.

### For CLI Users

1. Obtain a list of repository IDs:

```
# hammer repository list --organization "My_Organization"
```

2. Create the Content View and add repositories:

```
# hammer content-view create \  
--name "Example_Content_View" \  
--description "Example Content View" \  
--repository-ids 1,2 \  
--organization "My_Organization"
```

For the **--repository-ids** option, you can find the IDs in the output of the **hammer repository list** command.

3. Publish the view:

```
# hammer content-view publish \  
--name "Example_Content_View" \  
--description "Example Content View" \  
--organization "My_Organization"
```



Satellite Server creates the new version of the view and publishes it to the Library environment.

## 8.2. CREATING A CONTENT VIEW WITH A PUPPET MODULE

Use this procedure to create a Content View using one repository and no filters.

### Procedure

To create a Content View with a Puppet module, complete the following steps:

1. In the Satellite web UI, navigate to **Content** > **Content Views** and click **Create New View**.
2. In the **Name** field, enter a name for the view. Red Hat Satellite 6 automatically completes the **Label** field from the name you enter.
3. In the **Description** field, enter a description of the view.
4. Click **Save** to complete.
5. In the **Repository Selection** area, select the repositories that you want to add to your Content View, then click **Add Repositories**.
6. Click the **Puppet Modules** tab, then click **Add New Module**.
7. Search for the module that you want to add and click **Select a Version**.
8. Navigate to the entry for **Use Latest** and click **Select Version** in the **Actions** column.
9. To publish, click the **Versions** tab and click **Publish New Version**. In the **Description** field, enter a description to log the changes and click **Save**.

To register a host to your content view, see [Registering a Host](#) in the *Managing Hosts* guide.

## 8.3. PROMOTING A CONTENT VIEW

Use this procedure to promote Content Views across different lifecycle environments.

### Permission Requirements for Content View Promotion

Non-administrator users require two permissions to promote a Content View to an environment:

1. **promote\_or\_remove\_content\_views**
2. **promote\_or\_remove\_content\_views\_to\_environment**.

The **promote\_or\_remove\_content\_views** permission restricts which Content Views a user can promote.

The **promote\_or\_remove\_content\_views\_to\_environment** permission restricts the environments to which a user can promote Content Views.

With these permissions you can assign users permissions to promote certain Content Views to certain environments, but not to other environments. For example, you can limit a user so that they are permitted to promote to test environments, but not to production environments.

You must assign both permissions to a user to allow them to promote Content Views.

## Procedure

To promote a Content View, complete the following steps:

1. In the Satellite web UI, navigate to **Content** > **Content Views** and select the Content View that you want to promote.
2. Click the **Versions** tab for the Content View.
3. Select the version that you want to promote and in the **Actions** column, click **Promote**.
4. Select the environment where you want to promote the Content View, and click **Promote Version**.
5. Click the **Promote** button again. This time select the **Testing** environment and click **Promote Version**.
6. Finally click on the **Promote** button again. Select the **Production** environment and click **Promote Version**.

Now the repository for the Content View appears in all environments.

### For CLI Users

- Promote the Content View using the **hammer content-view version promote** each time:

```
# hammer content-view version promote \  
--content-view "Database" \  
--version 1 \  
--to-lifecycle-environment "Development" \  
--organization "My_Organization" \  
# hammer content-view version promote \  
--content-view "Database" \  
--version 1 \  
--to-lifecycle-environment "Testing" \  
--organization "My_Organization" \  
# hammer content-view version promote \  
--content-view "Database" \  
--version 1 \  
--to-lifecycle-environment "Production" \  
--organization "My_Organization"
```

Now the database content is available in all environments.

To register a host to your content view, see [Registering a Host](#) in the *Managing Hosts* guide.

## 8.4. COMPOSITE CONTENT VIEWS OVERVIEW

A Composite Content View combines the content from several Content Views. For example, you might have separate Content Views to manage an operating system and an application individually. You can use a Composite Content View to merge the contents of both Content Views into a new repository. The repositories for the original Content Views still exist but a new repository also exists for the combined content.

If you want to develop an application that supports different database servers. The *example\_application* appears as:

<i>example_software</i>
Application
Database
Operating System

Example of four separate Content Views:

- Red Hat Enterprise Linux (Operating System)
- PostgreSQL (Database)
- MariaDB (Database)
- *example\_software* (Application)

From the previous Content Views, you can create two Composite Content Views.

Example Composite Content View for a PostgreSQL database:

Composite Content View 1 - <i>example_software</i> on PostgreSQL
<i>example_software</i> (Application)
PostgreSQL (Database)
Red Hat Enterprise Linux (Operating System)

Example Composite Content View for a MariaDB:

Composite Content View 2 - <i>example_software</i> on MariaDB
<i>example_software</i> (Application)
MariaDB (Database)
Red Hat Enterprise Linux (Operating System)

Each Content View is then managed and published separately. When you create a version of the application, you publish a new version of the Composite Content Views. You can also select the **Auto Publish** option when creating a Composite Content View, and then the Composite Content View is automatically republished when a Content View it includes is republished.

## Repository Restrictions

You cannot include more than one of each repository in Composite Content Views. For example, if you attempt to include two Content Views using the same repository in a Composite Content View, Satellite Server reports an error.

## 8.5. CREATING A COMPOSITE CONTENT VIEW

### Procedure

To create a Composite Content View, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Content Views** and click **Create New View**.
2. In the **Name** field, enter a name for the view. Red Hat Satellite 6 automatically completes the **Label** field from the name you enter.
3. In the **Description** field, enter a description of the view.
4. Select the **Composite View?** check box to create a Composite Content View.
5. Optional: select the **Auto Publish** check box if you want the Composite Content View to be republished automatically when a Content View is republished.
6. Click **Save**.
7. In the **Add Content Views** area, select the Content Views that you want to add to the Composite Content View, and then click **Add Content Views**.
8. Click **Publish New Version** to publish the Composite Content View. In the **Description** field, enter a description and click **Save**.
9. Click **Promote** and select the lifecycle environments to promote the Composite Content View to, enter a description, and then click **Promote Version**.

### For CLI Users

1. Before you create the Composite Content Views, list the version IDs for our existing Content Views:

```
# hammer content-view version list \
--organization "My_Organization"
```

2. Create a new Composite Content View and include a comma-separated list of Content View Version IDs with the **--component-ids** option. When the **--auto-publish** option is set to **yes**, the Composite Content View is automatically republished when a Content View it includes is republished:

```
# hammer content-view create \
--composite \
--auto-publish yes \
--name "Example_Composite_Content_View" \
--description "Example Composite Content View" \
--component-ids Content_View_Version_IDs \
--organization "My_Organization"
```

3. Publish the Composite Content View:

■

```
# hammer content-view publish \
--name "Example_Composite_Content_View" \
--description "Initial version of Composite Content View" \
--organization "My_Organization"
```

- Promote the Composite Content View across all environments:

```
# hammer content-view version promote \
--content-view "Example_Composite_Content_View" \
--version 1 \
--to-lifecycle-environment "Development" \
--organization "My_Organization"
# hammer content-view version promote \
--content-view "Example_Composite_Content_View" \
--version 1 \
--to-lifecycle-environment "Testing" \
--organization "My_Organization"
# hammer content-view version promote \
--content-view "Example_Composite_Content_View" \
--version 1 \
--to-lifecycle-environment "Production" \
--organization "My_Organization"
```

## 8.6. CONTENT FILTER OVERVIEW

Content Views also use filters to include or restrict certain RPM content. Without these filters, a Content View includes everything from the selected repositories.

There are two types of content filters:

**Table 8.1. Filter Types**

Filter Type	Description
<b>Include</b>	You start with no content, then select which content to add from the selected repositories. Use this filter to combine multiple content items.
<b>Exclude</b>	You start with all content from selected repositories, then select which content to remove. Use this filter when you want to use most of a particular content repository but exclude certain packages, such as blacklisted packages. The filter uses all content in the repository except for the content you select.

### Include and Exclude Filter Combinations

If using a combination of Include and Exclude filters, publishing a Content View triggers the include filters first, then the exclude filters. In this situation, select which content to include, then which content to exclude from the inclusive subset.

### Content Types

There are also four types of content to filter:

Table 8.2. Content Types

Content Type	Description
Package	Filter packages based on their name and version number.
Package Group	Filter packages based on package groups. The list of package groups is based on the repositories added to the Content View.
Erratum (by ID)	Select which specific errata to add to the filter. The list of Errata is based on the repositories added to the Content View.
Erratum (by Date and Type)	Select a issued or updated date range and errata type (Bugfix, Enhancement, or Security) to add to the filter.

## Resolving Package Dependencies and Filters

Filters do not resolve any dependencies of the packages listed in the filters. Ensure to add package dependencies to the filter. This might require some level of testing to determine what dependencies are required.

## 8.7. CONTENT FILTER EXAMPLES

Use any of the following examples with the procedure that follows to build custom content filters.

### Example 1

Create a repository with the base Red Hat Enterprise Linux packages. This filter requires a Red Hat Enterprise Linux repository added to the Content View.

#### Filter:

- **Inclusion Type:** Include
- **Content Type:** Package Group
- **Filter:** Select only the **Base** package group

### Example 2

Create a repository that excludes all errata, except for security updates, after a certain date. This is useful if you want to perform system updates on a regular basis with the exception of critical security updates, which must be applied immediately. This filter requires a Red Hat Enterprise Linux repository added to the Content View.

#### Filter:

- **Inclusion Type:** Exclude
- **Content Type:** Erratum (by Date and Type)

- **Filter:** Select only the **Bugfix** and **Enhancement** errata types, and clear the **Security** errata type. Set the **Date Type** to **Updated On**. Set the **Start Date** to the date you want to restrict errata. Leave the **End Date** blank to ensure any new non-security errata is filtered.

### Example 3

A combination of Example 1 and Example 2 where you only require the operating system packages and want to exclude recent bug fix and enhancement errata. This requires two filters attached to the same Content View. The Content View processes the Include filter first, then the Exclude filter.

#### Filter 1:

- **Inclusion Type:** Include
- **Content Type:** Package Group
- **Filter:** Select only the **Base** package group

#### Filter 2:

- **Inclusion Type:** Exclude
- **Content Type:** Erratum (by Date and Type)
- **Filter:** Select only the **Bugfix** and **Enhancement** errata types, and clear the **Security** errata type. Set the **Date Type** to **Updated On**. Set the **Start Date** to the date you want to restrict errata. Leave the **End Date** blank to ensure any new non-security errata is filtered.

For another example of how content filters work, see the following article: "[How do content filters work in Satellite 6](#)"

## 8.8. CREATING A CONTENT FILTER

Use this procedure to create a content filter. For examples of how to build a filter, see [Section 8.7, "Content Filter Examples"](#)

### Procedure

To create a content filter, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Content Views** and select a Content View.
2. Navigate to **Yum Content > Filters** and click **New Filter**.
3. In the **Name** field, enter a name for your filter.
4. From the **Content Type** list, select the content type that you want to filter. Depending on what you select for the new filter's content type, different options appear.
5. From the **Inclusion Type** list, select either **Include** or **Exclude**.
6. In the **Description** field, enter a description for the filter, and click **Save**.
7. Depending on what you enter for **Content Type**, add rules to create the filter that you want.
8. Click the **Affected repositories** tab to select which specific repositories use this filter.

- Click **Publish New Version** to publish the filtered repository. In the **Description** field, enter a description of the changes, and click **Save**.

You can promote this Content View across all environments.

### For CLI Users

- Add a filter to the Content View. Use the **--inclusion false** option to set the filter to an Exclude filter:

```
# hammer content-view filter create \  
--name "Errata Filter" \  
--type erratum --content-view "Example_Content_View" \  
--description "My latest filter" \  
--inclusion false \  
--organization "My_Organization"
```

- Add a rule to the filter:

```
# hammer content-view filter rule create \  
--content-view "Example_Content_View" \  
--content-view-filter "Errata Filter" \  
--start-date "YYYY-MM-DD" \  
--types enhancement,bugfix \  
--date-type updated \  
--organization "My_Organization"
```

- Publish the Content View:

```
# hammer content-view publish \  
--name "Example_Content_View" \  
--description "Adding errata filter" \  
--organization "My_Organization"
```

- Promote the view across all environments:

```
# hammer content-view version promote \  
--content-view "Example_Content_View" \  
--version 1 \  
--to-lifecycle-environment "Development" \  
--organization "My_Organization" \  
# hammer content-view version promote \  
--content-view "Example_Content_View" \  
--version 1 \  
--to-lifecycle-environment "Testing" \  
--organization "My_Organization" \  
# hammer content-view version promote \  
--content-view "Example_Content_View" \  
--version 1 \  
--to-lifecycle-environment "Production" \  
--organization "My_Organization"
```



## CHAPTER 9. MANAGING ACTIVATION KEYS

Activation keys provide a method to automate system registration and subscription attachment. You can create multiple keys and associate them with different environments and Content Views. For example, you might create a basic activation key with a subscription for Red Hat Enterprise Linux workstations and associate it with Content Views from a particular environment.

You can use activation keys during content host registration to improve the speed, simplicity and consistency of the process.

Activation keys can define the following properties for content hosts:

- Associated subscriptions and subscription attachment behavior.
- Available products and repositories.
- A life cycle environment and a Content View.
- Host collection membership.

You can apply the same activation key to multiple content hosts if it contains enough subscriptions. However, activation keys set only the initial configuration for a content host. When the content host is registered to an organization, the organization's content can be attached to the content host manually.

A content host can be associated with multiple activation keys that are combined to define the host settings. In case of conflicting settings, the last specified activation key takes precedence.

Note that activation keys are used only when hosts are registered. If changes are made to an activation key, it is applicable only to hosts that are registered with the amended activation key in the future. The changes are not made to existing hosts.

### Content View Conflicts between Host Creation and Registration

When you provision a host, Satellite uses provisioning templates and other content from the Content View that you set in the host group or host settings. When the host is registered, the Content View from the activation key overwrites the original Content View from the host group or host settings. Then Satellite uses the Content View from the activation key for every future task, for example, rebuilding a host.

When you rebuild a host, ensure that you set the Content View that you want to use in the activation key and not in the host group or host settings.

## 9.1. CREATING AN ACTIVATION KEY

With activation keys, you can define how the content host is subscribed during registration. The subscription behavior defined by the activation key depends on two factors:

- Are there any subscriptions associated with the activation key?
- Is the auto-attach option enabled?

Based on the previous factors, there are three possible scenarios for subscribing with activation keys:

1. Activation key with no subscriptions specified.

With no subscriptions specified and auto-attach enabled, hosts using the activation key search for the best fitting subscription from the ones provided by Satellite Server. This is similar to entering the **subscription-manager --auto-attach** command.

2. Activation key providing a custom subscription pool for auto-attach.  
If there are subscriptions specified and auto-attach is enabled, hosts using the activation key select the best fitting subscription from the list specified in the activation key.
3. Activation key with the exact set of subscriptions.  
If there are subscriptions specified and auto-attach is disabled, hosts using the activation key are associated with all subscriptions specified in the activation key.

## Custom Products

If a custom product, typically containing content not provided by Red Hat, is assigned to an activation key, this product is always enabled for the registered content host regardless of the auto-attach setting.

## Procedure

To create an activation key, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Activation keys** and click **Create Activation Key**.
2. In the **Name** field, enter the name of the activation key.
3. If you want to set a limit, clear the **Unlimited hosts** check box, and in the **Limit** field, enter the maximum number of systems you can register with the activation key. If you want unlimited hosts to register with the activation key, ensure the **Unlimited Hosts** check box is selected.
4. In the **Description** field, enter a description for the activation key.
5. From the **Environment** list, select the environment to use.
6. From the **Content View** list, select a Content View to use.
7. Click **Save** and when your new activation key appears in the Activation Keys window, click the name to edit.
8. To define which products to attach and repositories to enable upon registration, click the **Subscriptions** tab, select the subscriptions that you want to add, then click **Add**.
9. In the Satellite web UI, navigate to the **Product Content** page. For example, to enable the Red Hat Satellite Tools repository, for **Red Hat Satellite Tools 6.4 (for RHEL 7 Server) (RPMs) - Enabled** select **Override to Yes**.
10. Click **Save**.

## For CLI Users

1. Create the activation key:

```
# hammer activation-key create \  
--name "My_Activation_Key" \  
--unlimited-hosts \  
--description "Example Stack in the Development Environment" \  
--lifecycle-environment "Development" \  
--content-view "Stack" \  
--organization "My_Organization"
```

2. Obtain a list of your subscription IDs:

```
# hammer subscription list --organization "My_Organization"
```

3. Attach the Red Hat Enterprise Linux subscription UUID to the activation key:

```
# hammer activation-key add-subscription \
--name "My_Activation_Key" \
--subscription-id ff808181533518d50152354246e901aa \
--organization "My_Organization"
```

4. List the product content associated with the activation key:

```
# hammer activation-key product-content \
--name "My_Activation_Key" \
--organization "My_Organization"
```

5. Override the default auto-enable status for the Red Hat Satellite Tools 6.4 repository. The default status is set to disabled. To enable, enter the following command:

```
# hammer activation-key content-override \
--name "My_Activation_Key" \
--content-label rhel-7-server-satellite-tools-6.4-rpms \
--value 1 \
--organization "My_Organization"
```

## 9.2. UPDATING SUBSCRIPTIONS ASSOCIATED WITH AN ACTIVATION KEY

You can change the subscriptions associated with an activation key using the web UI or using the Hammer command-line tool.

Note that changes to an activation key apply only to machines provisioned after the change. To update subscriptions on existing content hosts, see [Section 4.2.6, "Bulk Updating Content Hosts' Subscriptions"](#).

### Procedure

To update the subscriptions associated with an activation key, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Activation keys** and click the name of the activation key.
2. Click the **Subscriptions** tab.
3. To remove subscriptions, select **List/Remove**, and then select the check boxes to the left of the subscriptions to be removed and then click **Remove Selected**.
4. To add subscriptions, select **Add**, and then select the check boxes to the left of the subscriptions to be added and then click **Add Selected**.
5. Click the **Repository Sets** tab and review the repositories' status settings.
6. To enable or disable a repository, select the check box for a repository and then change the status using the **Select Action** list.

- Click the **Details** tab, select a Content View for this activation key, and then click **Save**.

### For CLI Users

- List the subscriptions that the activation key currently contains:

```
# hammer activation-key subscriptions \
--name My_Activation_Key \
--organization "My_Organization"
```

- Remove the required subscription from the activation key:

```
# hammer activation-key remove-subscription \
--name "My_Activation_Key" \
--subscription-id ff808181533518d50152354246e901aa \
--organization "My_Organization"
```

For the **--subscription-id** option, you can use either the UUID or the ID of the subscription.

- Attach new subscription to the activation key:

```
# hammer activation-key add-subscription \
--name "My_Activation_Key" \
--subscription-id ff808181533518d50152354246e901aa \
--organization "My_Organization"
```

For the **--subscription-id** option, you can use either the UUID or the ID of the subscription.

- List the product content associated with the activation key:

```
# hammer activation-key product-content \
--name "My_Activation_Key" \
--organization "My_Organization"
```

- Override the default auto-enable status for the required repository:

```
# hammer activation-key content-override \
--name "My_Activation_Key" \
--content-label content_label \
--value 1 \
--organization "My_Organization"
```

For the **--value** option, enter **1** for enable, **0** for disable.

## 9.3. UPDATING SUBSCRIPTIONS ASSOCIATED WITH AN ACTIVATION KEY USING A CSV FILE

You can update the subscriptions associated with an activation key by downloading a CSV file, making changes to the Activation Key settings, and then uploading the changed CSV file.

### Subscription information in the CSV File

In the CSV file, the last column contains the subscription information. Subscription information is one field in the CSV file and can include a comma in quoted text.

The entries in the CSV file have the following format:

```
Name Organization Description Limit Environment Content View Host Collections Auto-Attach
Service Level Release Version Subscriptions
```

Examples of the **Subscriptions** field include:

```
Automatic|RH1234|Red Hat Enterprise Linux Server, Standard (Physical or Virtual
Nodes)|11223344|55667788
```

```
1|MCT0369|Red Hat Satellite Capsule Server|11223344|55667788
```

The **Subscriptions** field has the following format:

- Number of subscriptions allocated. This can be set to **Automatic**.
- The subscription's identification number.
- The subscription's name.
- The contract number.
- The account number.

## Procedure

To update the subscription list using a CSV file, complete the following procedure:

1. Export the subscriptions from Satellite Server to a CSV file. For this example, **a\_keys.csv**:

```
# hammer --server https://satellite.example.com csv activation-keys \
--export --file a_keys.csv --organization "My_Organization"
```

2. To view the columns of the file, enter the following command:

```
# column -s, -t < a_keys.csv | less -S
```

3. Change the required values in the CSV file. You can use an editor, with a CSV plug-in, or **sed** to change strings in the **Subscriptions** field.
4. Make a backup of the file.

```
# cp a_keys.csv a_keys.csv.backup
```

5. Edit the string you want to change. For example:

```
# sed -i "s/Automatic|RH1234|Red Hat Enterprise Linux Server/Automatic|RH4567|Red Hat
Enterprise Linux Server/g" a_keys.csv
```

6. Confirm only the required changes were made. For example:

```
# diff a_keys.csv a_keys.csv.backup
```

7. Upload the changed file to Satellite Server.

```
# hammer --server https://satellite.example.com csv activation-keys \  
--file a_keys.csv
```

## 9.4. USING ACTIVATION KEYS FOR HOST REGISTRATION

You can use activation keys to complete the following tasks:

- Registering new hosts during provisioning through Red Hat Satellite 6. The kickstart provisioning templates in Red Hat Satellite 6 contain commands to register the host using an activation key that is defined when creating a host.
- Registering existing Red Hat Enterprise Linux hosts. Configure Red Hat Subscription Manager to use Satellite Server for registration and specify the activation key when running the **subscription-manager register** command.

### Procedure

To use an activation key for host registration with an existing Red Hat Enterprise Linux 7 host to Satellite Server, complete the following steps:

1. Download the consumer RPM for your Satellite Server. This is located in the **pub** directory on the host's web server. For example, for a Satellite Server with the host name **satellite.example.com**, enter the following command on the host to register:

```
# rpm -Uvh http://satellite.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

This RPM installs the necessary certificates for accessing repositories on Satellite Server and configures Red Hat Subscription Manager to use the server's URL.

2. On the host, enter the following command to register the host to Satellite using the activation key:

```
# subscription-manager register --activationkey="My_Activation_Key" \  
--org="My_Organization"
```

3. To view a list of hosts for an organization, on Satellite Server, enter the following command:

```
# hammer host list --organization "My_Organization"
```

4. After registering a host to Satellite Server, install the **katello-agent** package on the host so that it can report back to Satellite Server:

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

The Red Hat Satellite Tools 6.4 repository provides this package.

### Multiple Activation Keys

You can use multiple activation keys when registering a content host. You can then create activation keys for specific subscription sets and combine them according to content host requirements. For example, the following command registers a content host to your organization with both VDC and OpenShift subscriptions:

```
# subscription-manager register --org="My_Organization" \
--activationkey="ak-VDC,ak-OpenShift"
```

## Settings Conflicts

If there are conflicting settings in activation keys, the rightmost key takes precedence.

- Settings that conflict: **Service Level**, **Release Version**, **Environment**, **Content View**, and **Product Content**.
- Settings that do not conflict and the host gets the union of them: **Subscriptions** and **Host Collections**.
- Settings that influence the behavior of the key itself and not the host configuration: **Content Host Limit** and **Auto-Attach**.

## 9.5. ENABLING AUTO-ATTACH

When auto-attach is enabled on an activation key and there are subscriptions associated with the key, the subscription management service selects and attaches the best-matched associated subscriptions based on a set of criteria like currently-installed products, architecture, and preferences like service level.

You can enable auto-attach and have no subscriptions associated with the key. This type of key is commonly used to register virtual machines when you do not want the virtual machine to consume a RHEL subscription but to inherit a RHEL Virtual Data Center (VDC) subscription from the hypervisor.

Auto-attach is enabled by default. Disable the option if you want to force attach all subscriptions associated with the activation key.

### Procedure

To enable auto-attach, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Activation Keys**.
2. Click the activation key name that you want to edit.
3. Click the **Subscriptions** tab.
4. Click the edit icon next to **Auto-Attach**.
5. Select or clear the check box to enable or disable auto-attach.
6. Click **Save**.

To register virtual content hosts to Satellite Server using an auto-attach activation key, first use the **virt-who** utility to map those hosts to a hypervisor entitled with the Virtual Datacenter (VDC) subscription. Without this prerequisite, virtual hosts are registered only with a temporary virtual subscription for 24 hours. For more information, see [Applying Virtual Guest Subscriptions](#) in the *Virtual Instances Guide*.

### For CLI Users

To enable auto-attach on the activation key:

```
# hammer activation-key update --name "My_Activation_Key" \
--organization "My_Organization" --auto-attach true
```

-

## 9.6. SETTING THE SERVICE LEVEL

You can configure an activation key to define a default service level for the new host created with the activation key. Setting a default service level selects only the matching subscriptions to be attached to the host. For example, if the default service level on an activation key is set to Premium, only subscriptions with premium service levels are attached to the host upon registration.

### Procedure

To set the service level, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Activation Keys**.
2. Click the activation key name you want to edit.
3. Click the edit icon next to **Service Level**.
4. Select the required service level from the list. The list only contains service levels available to the activation key.
5. Click **Save**.

### For CLI Users

To set a default service level to Premium on the activation key:

```
# hammer activation-key update --name "My_Activation_Key" \  
--organization "My_Organization" --service-level premium
```



## CHAPTER 10. MANAGING ERRATA

As a part of Red Hat’s quality control and release process, we provide customers with updates for each release of official Red Hat RPMs. Red Hat compiles groups of related package into an **erratum** along with an advisory that provides a description of the update. There are three types of advisories (in order of importance):

### Security Advisory

Describes fixed security issues found in the package. The security impact of the issue can be Low, Moderate, Important, or Critical.

### Bug Fix Advisory

Describes bug fixes for the package.

### Product Enhancement Advisory

Describes enhancements and new features added to the package.

Red Hat Satellite 6 imports this errata information when synchronizing repositories with Red Hat’s Content Delivery Network (CDN). Red Hat Satellite 6 also provides tools to inspect and filter errata, allowing for precise update management. This way, you can select relevant updates and propagate them through Content Views to selected content hosts.

Errata are labeled according to the most important advisory type they contain. Therefore, errata labeled as **Product Enhancement Advisory** can contain only enhancement updates, while **Bug Fix Advisory** errata can contain both bug fixes and enhancements, and **Security Advisory** can contain all three types.

In Red Hat Satellite, there are two keywords that describe an erratum’s relationship to the available content hosts:

### Applicable

Erratum applies to one or more content hosts, which means it updates packages present on the content host. Applicable errata are not yet accessible by the content host.

### Installable

Erratum applies to one or more content hosts and it has been made available to the content host. Installable errata are present in the content host’s life cycle environment and Content View, but are not yet installed. This way, errata can be installed by users who have permissions to manage content hosts, but are not entitled for errata management at higher levels.

This chapter shows how to manage errata and apply them to either a single system or multiple systems.

## 10.1. INSPECTING AVAILABLE ERRATA

The following procedure describes how to view and filter the available errata and how to display metadata of the selected advisory.

1. Navigate to **Content > Errata** to view the list of available errata.
2. Use the filtering tools at the top of the page to limit the number of displayed errata:
  - Select the repository to be inspected from the list. **All Repositories** is selected by default.
  - The **Applicable** check box is selected by default to view only errata applicable to the selected repository. Select the **Installable** check box to view only errata marked as installable.

- To search the table of errata, type the query in the **Search** field in the form of:

```
parameter operator value
```

See [Table 10.1, "Parameters Available for Errata Search"](#) for the list of parameters available for search. Find the list of applicable operators in [Supported Operators for Granular Search](#) in *Administering Red Hat Satellite*. Automatic suggestion works as you type. You can also combine queries with the use of **and** and **or** operators. For example, to display only security advisories related to the **kernel** package, type:

```
type = security and package_name = kernel
```

Press **Enter** to start the search.

- Click the **Errata ID** of the erratum you want to inspect:

- The **Details** tab contains the description of the updated package as well as documentation of important fixes and enhancements provided by the update.
- On the **Content Hosts** tab, you can apply the erratum to selected content hosts as described in [Section 10.5, "Applying Errata to Multiple Systems"](#).
- The **Repositories** tab lists repositories that already contain the erratum. You can filter repositories by the environment and Content View, and search for them by the repository name.

**Table 10.1. Parameters Available for Errata Search**

Parameter	Description	Example
bug	Search by the Bugzilla number.	<i>bug = 1172165</i>
cve	Search by the CVE number.	<i>cve = CVE-2015-0235</i>
id	Search by the errata ID. The auto-suggest system displays a list of available IDs as you type.	<i>id = RHBA-2014:2004</i>
issued	Search by the issue date. You can specify the exact date, like "Feb16,2015", or use keywords, for example "Yesterday", or "1 hour ago". The time range can be specified with the use of the "<" and ">" operators.	<i>issued &lt; "Jan 12,2015"</i>
package	Search by the full package build name. The auto-suggest system displays a list of available packages as you type.	<i>package = glib2-2.22.5-6.el6.i686</i>

Parameter	Description	Example
package_name	Search by the package name. The auto-suggest system displays a list of available packages as you type.	<i>package_name = glib2</i>
severity	Search by the severity of the issue fixed by the security update. Specify <i>Critical</i> , <i>Important</i> , or <i>Moderate</i> .	<i>severity = Critical</i>
title	Search by the advisory title.	<i>title ~ openssl</i>
type	Search by the advisory type. Specify <i>security</i> , <i>bugfix</i> , or <i>enhancement</i> .	<i>type = bugfix</i>
updated	Search by the date of the last update. You can use the same formats as with the <b>issued</b> parameter.	<i>updated = "6 days ago"</i>

## 10.2. SUBSCRIBING TO ERRATA NOTIFICATIONS

You can configure email notifications for Satellite users. Users receive a summary of applicable and installable errata, notifications on Content View promotion or after synchronizing a repository. For more information, see the [Configuring Email Notifications](#) section in the *Administering Red Hat Satellite* guide.

## 10.3. MANAGING ERRATA WITH CONTENT VIEWS

Red Hat Satellite 6 provides various methods to manage and apply errata. You can use Content Views and content filters to limit errata. Such filters include:

- **ID** - Select specific erratum to allow into your resulting repositories.
- **Date Range** - Define a date range and include a set of errata released during that date range.
- **Type** - Select the type of errata to include such as bug fixes, enhancements, and security updates.

### 10.3.1. Creating a Content View Filter for Errata

Create a content filter to exclude errata after a certain date. This ensures your production systems in the application life cycle are kept up to date to a certain point. Then you can modify the filter's start date to introduce new errata into your testing environment to test the compatibility of new packages into your application life cycle.

#### Prerequisites

- A Content View with the repositories that contain required errata is created. For more information, see [Section 8.1, "Creating a Content View"](#).

## Procedure

1. In the Satellite web UI, navigate to **Content** > **Content Views** and select a Content View that you want to use for applying errata.
2. Navigate to **Yum Content** > **Filters** and click **New Filter**.
3. In the **Name** field, enter **Errata Filter**.
4. From the **Content Type** list, select **Erratum - Date and Type**.
5. From the **Inclusion Type** list, select **Exclude**.
6. In the **Description** field, enter **Exclude errata items from YYYY-MM-DD**.
7. Click **Save**.
8. For **Errata Type**, select the check boxes of errata types you want to exclude. For example, select the **Enhancement** and **Bugfix** check boxes and clear the **Security** check box to exclude enhancement and bugfix errata after certain date, but include all the security errata.
9. For **Date Type**, select one of two check boxes:
  - **Issued On** for the issued date of the erratum.
  - **Updated On** for the date of the erratum's last update.
10. Select the **Start Date** to exclude all errata on or after the selected date.
11. Leave the **End Date** field blank.
12. Click **Save**.
13. Click **Publish New Version** to publish the resulting repository.
14. Enter **Adding errata filter** in the **Description** field.
15. Click **Save**.

When the Content View completes publication, notice the **Content** column reports a reduced number of packages and errata from the initial repository. This means the filter successfully excluded the all non-security errata from the last year.
16. Click the **Versions** tab.
17. Click **Promote** to the right of the published version.
18. Select the environments you want to promote the Content View version to.
19. In the **Description** field, enter the description for promoting.
20. Click **Promote Version** to promote this Content View version across the required environments.

## For CLI Users

1. Create a filter for the errata:

```
# hammer content-view filter create --name "Filter Name" \
--description "Exclude errata items from the YYYY-MM-DD" \
--content-view "CV Name" --organization "Default Organization" \
--type "erratum"
```

2. Create a filter rule to exclude all errata on or after the *Start Date* that you want to set:

```
# hammer content-view filter rule create --start-date "YYYY-MM-DD" \
--content-view "CV Name" --content-view-filter="Filter Name" \
--organization "Default Organization" --types=security,enhancement,bugfix
```

3. Publish the Content View:

```
# hammer content-view publish --name "CV Name" \
--organization "Default Organization"
```

4. Promote the Content View to the lifecycle environment so that the included errata are available to that lifecycle environment:

```
# hammer content-view version promote \
--content-view "CV Name" \
--organization "Default Organization" \
--to-lifecycle-environment "Lifecycle Environment Name"
```

## 10.4. APPLYING ERRATA TO INDIVIDUAL SYSTEMS

Use these procedures to review and apply errata to individual systems.

### Prerequisites

- Synchronize Red Hat Satellite repositories with the latest errata available from Red Hat. For more information, see [Section 5.6, "Synchronizing Red Hat Repositories"](#) .
- Register the host to an environment and Content View on Satellite Server. For more information, see [Registering a Host](#) in the *Managing Hosts* guide.
- Install the **katello-agent** package on the host. For more information, see the [Installing the Katello Agent](#) section in the *Managing Hosts* guide.

### Procedure

1. In the Satellite web UI, navigate to **Hosts** > **Content Hosts** and select the host you want to apply errata to.
2. Navigate to the **Errata** tab to see the list of errata.
3. Select the errata to apply and click **Apply Selected**. In the confirmation window, click **Apply**.
4. After the task to update all packages associated with the selected errata completes, click the **Details** tab to view the updated packages.

## For CLI Users

1. List all errata for the host:

```
# hammer host errata list \
--host client.example.com
```

2. Apply the most recent erratum to the host. Identify the erratum to apply using the erratum ID:

```
# hammer host errata apply --host "Host Name" \
--errata-ids ERRATUM_ID1,ERRATUM_ID2...
```

## 10.5. APPLYING ERRATA TO MULTIPLE SYSTEMS

Use these procedures to review and apply errata to multiple systems.

### Prerequisites

- Synchronize Red Hat Satellite repositories with the latest errata available from Red Hat. For more information, see [Section 5.6, "Synchronizing Red Hat Repositories"](#).
- Register the hosts to an environment and Content View on Satellite Server. For more information, see [Registering a Host](#) in the *Managing Hosts* guide.
- Install the **katello-agent** package on hosts. For more information, see the [Installing the Katello Agent](#) section in the *Managing Hosts* guide.

### Procedure

1. Navigate to **Content > Errata**.
2. Click the name of an erratum you want to apply.
3. Click to **Content Hosts** tab.
4. Select the systems you want to apply errata to and click **Apply to Hosts**.
5. Click **Confirm**.

### For CLI Users

Although the CLI does not have the same tools as the Web UI, you can replicate a similar procedure with CLI commands.

1. List all installable errata:

```
# hammer erratum list \
--errata-restrict-installable true \
--organization "Default Organization"
```

2. Select the erratum you want to use and list the systems that this erratum is applicable to:

```
# hammer host list \
--search "applicable_errata = ERRATUM_ID" \
--organization "Default Organization"
```

- 
3. Apply the errata to a single system:

```
# hammer host errata apply \  
--host client.example.com \  
--organization "Default Organization" \  
--errata-ids ERRATUM_ID1,ERRATUM_ID2...
```

4. Enter the following command for each host and replace **\$HOST** with the name of the system for each execution.

```
# for HOST in `hammer \  
--csv --csv-separator "|" host list \  
--search "applicable_errata = ERRATUM_ID" \  
--organization "Default Organization" | tail -n+2 | awk \  
-F "|" '{ print $2 }' ; do echo \  
"== Applying to $HOST ==" ; hammer host errata apply \  
--host $HOST --errata-ids ERRATUM_ID1,ERRATUM_ID2 ; done
```

This command identifies all hosts with *erratum\_IDs* as an applicable erratum and then applies the erratum to each host.

## CHAPTER 11. MANAGING OSTREE CONTENT

**OSTree** is a tool to manage bootable, immutable, versioned file system trees. You can use a custom OSTree content on a build system, then export an OSTree repository to a static HTTP. Red Hat Enterprise Linux Atomic Server uses OSTree content composed from RPM files as a method to keep the operating system up to date.

You can use Red Hat Satellite 6 to synchronize and manage OSTree branches from an OSTree repository.

In Satellite Server 6.4, OSTree management tools are enabled by default. If you ever have a reason to enable the tool, enter the following command:

```
# satellite-installer --katello-enable-ostree=true
```

### 11.1. SELECTING RED HAT OSTREE CONTENT TO SYNCHRONIZE

Red Hat CDN provides OSTree Content for you to select and synchronize.

#### Procedure

To find and synchronize OSTree content, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Red Hat Repositories**.
2. From the list, select the **OSTree** content type.
3. In the Available Repositories pane, locate the OSTree repository set you want to use, for example, the **Red Hat Enterprise Linux Atomic Host Treesset** from the **Red Hat Enterprise Linux Atomic Host** product group.
4. Click the **Enable** icon to enable the repository you want to use.
5. Navigate to **Content > Products** and click the product that you want to use, for example **Red Hat Enterprise Linux Atomic Host**.
6. Select the upstream synchronization policy for this repository. By default, Satellite synchronizes only the latest OSTree branch.
  - a. Click the repository you want to synchronize.
  - b. From the **Upstream Sync Policy** menu, select one of the following policies to synchronize OSTree branches for this repository:
    - **Latest Only** - synchronize only the latest OSTree branch.
    - **All History** - synchronize all OSTree branches.
    - **Custom** - synchronize a custom number of OSTree branches. Enter the required number into the field below.
  - c. Click **Save**.
7. From the **Select Action** menu, select **Sync Now**.

**To view the Synchronization Status**



- In the Satellite web UI, navigate to **Content > Sync Status** and expand, for example, **Red Hat Enterprise Linux Atomic Host**.

### For CLI Users

1. Search the Red Hat Enterprise Linux Server product for **ostree** repositories:

```
# hammer repository-set list \
--product "Red Hat Enterprise Linux Atomic Host" \
--organization "My_Organization" | grep "ostree"
```

2. Enable the **ostree** repository for Red Hat Enterprise Linux Atomic Host or any product that you want to use:

```
# hammer repository-set enable \
--product "Red Hat Enterprise Linux Atomic Host" \
--name "Red Hat Enterprise Linux Atomic Host (Trees)" \
--organization "My_Organization"
```

3. Locate and synchronize the repository for the product:

```
# hammer repository list \
--product "Red Hat Enterprise Linux Atomic Host" \
--organization "My_Organization"
# hammer repository synchronize \
--name "Red Hat Enterprise Linux Atomic Host Trees" \
--product "Red Hat Enterprise Linux Atomic Host" \
--organization "My_Organization"
```

## 11.2. IMPORTING CUSTOM OSTREE CONTENT

In addition to importing OSTree content from Red Hat CDN, you can also import content from other sources. This requires a published HTTP location for the OSTree to import.

### Procedure

To import custom OSTree content, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Products** and click **Create Product**.
2. In the **Name** field, enter a name for your OSTree content. This automatically populates the **Label** field.
3. Optional: In the **GPG Key** field, enter a GPG Key for the entire product.
4. From the **Sync Plan** menu, select a synchronization plan to associate with the product.
5. In the **Description** field, enter a description of the product and click **Save**.
6. When the product creation completes, click **Create Repository**.
7. In the **Name** field, enter a name for the repository. This automatically populates the **Label** field.
8. From the **Type** list, select **ostree**.

9. In the **URL** field, enter the URL of the registry to use as a source. For example <http://www.example.com/rpm-ostree/>.
10. From the **Upstream Sync Policy** menu, select one of the following policies to synchronize OSTree branches for this repository:
  - **Latest Only** - synchronize only the latest OSTree branch.
  - **All History** - synchronize all OSTree branches.
  - **Custom** - synchronize a custom number of OSTree branches. Enter the required number into the field below.
11. Click **Save**.
12. When the repository creation completes, select the new repository and click **Sync Now** to start the synchronization process.

#### To view the Synchronization Status:

- In the Satellite web UI, navigate to **Content > Sync Status** and expand the entry that you want to view.

#### For CLI Users

1. Create the custom **OSTree Content** product:

```
# hammer product create \
--name "Custom OSTree Content" \
--sync-plan "Example_Plan" \
--description "OSTree Content" \
--organization "My_Organization"
```

2. Create the repository for the OSTree:

```
# hammer repository create \
--name "Custom OSTree" \
--content-type "ostree" \
--url "http://www.example.com/rpm-ostree/" \
--product "OSTree Content" \
--organization "My_Organization"
```

3. Synchronize the repository:

```
# hammer repository synchronize \
--name "Custom OSTree" \
--product "OSTree Content" \
--organization "My_Organization"
```

## 11.3. MANAGING OSTREE CONTENT WITH CONTENT VIEWS

Use Content Views to manage OSTree branches across the application life cycle. This process uses the same publication and promotion method that RPMs and Puppet modules use.

## Procedure

To create a content view for your OSTree and add a repository, complete the following steps:

1. In the Satellite web UI, navigate to **Content** > **Content Views** and click **Create New View**.
2. In the **Name** field, enter a plain text name for the view. This automatically populates the **Label** field.
3. In the **Description** field, enter a description of the OSTree Content View.
4. If you want to use a Composite Content View, select the **Composite View** check box.
5. Click **Save** to complete.
6. Navigate to the **OSTree Content** tab, then click **Add**.
7. Select the OSTree repository for that you want to use. Click **Add Repository** to add the OSTree content from this repository to the Content View.
8. Navigate to **Versions** and click **Publish New Version**.
9. In the **Description** field, enter a description for the version, and click **Save**.

You can also click **Promote** to promote this Content View across environments in the application life cycle.

## For CLI Users

1. Obtain a list of repository IDs:

```
# hammer repository list --organization "_My_Organization_"
```

2. Create the Content View and add the repository:

```
# hammer content-view create \
--name "OSTree" \
--description "OSTree for Red Hat Enterprise Linux Atomic Host" \
--repository-ids 5 \
--organization "My_Organization"
```

3. Publish the view:

```
# hammer content-view publish \
--name "OSTree" \
--description "Example Content View for the OSTree" \
--organization "My_Organization"
```

## CHAPTER 12. MANAGING ISO IMAGES

You can use Red Hat Satellite 6 to store ISO images, either from Red Hat's Content Delivery Network or other sources. You can also upload other files, such as virtual machine images, and publish them in repositories.

### 12.1. IMPORTING ISO IMAGES FROM RED HAT

The Red Hat Content Delivery Network provides ISO images for certain products. The procedure for importing this content is similar to the procedure for enabling repositories for RPM content.

#### Procedure

To import Red Hat ISO images, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Red Hat Repositories**.
2. In the **Search** field, enter an image name, for example, **Red Hat Enterprise Linux 7 Server (ISOs)**.
3. In the Available Repositories window, expand **Red Hat Enterprise Linux 7 Server (ISOs)**
4. For the **x86\_64 7.2** entry, click the **Enable** icon to enable the repositories for the image.
5. Navigate to **Content > Products** and click **Red Hat Enterprise Linux Server**.
6. Click the **Repositories** tab of the Red Hat Enterprise Linux Server window, and click **Red Hat Enterprise Linux 7 Server ISOs x86\_64 7.2**.
7. In the upper right of the Red Hat Enterprise Linux 7 Server ISOs x86\_64 7.2 window, click **Select Action** and select **Sync Now**.

#### To view the Synchronization Status

- In the web UI, navigate to **Content > Sync Status** and expand **Red Hat Enterprise Linux Server**.

#### For CLI Users

1. Locate the Red Hat Enterprise Linux Server product for **file** repositories:

```
# hammer repository-set list \
--product "Red Hat Enterprise Linux Server" \
--organization "My_Organization" | grep "file"
```

2. Enable the **file** repository for Red Hat Enterprise Linux 7.2 Server ISO:

```
# hammer repository-set enable \
--product "Red Hat Enterprise Linux Server" \
--name "Red Hat Enterprise Linux 7 Server (ISOs)" \
--releasever 7.2 \
--basearch x86_64 \
--organization "My_Organization"
```

3. Locate and synchronize the repository in the product:

```
# hammer repository list \
--product "Red Hat Enterprise Linux Server" \
--organization "My_Organization"
# hammer repository synchronize \
--name "Red Hat Enterprise Linux 7 Server ISOs x86_64 7.2" \
--product "Red Hat Enterprise Linux Server" \
--organization "My_Organization"
```

## 12.2. IMPORTING INDIVIDUAL ISO IMAGES AND FILES

Use this procedure to manually import ISO content and other files to Satellite Server. To import custom files, you can complete the following steps in the web UI or using the Hammer CLI. However, if the size of the file that you want to upload is larger than 15 MB, you must use the Hammer CLI to upload it to a repository.

1. Create a custom product.
2. Add a repository for files to the product.
3. Upload a file to the repository.

### Procedure

To import custom ISO images, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Products**, and in the Products window, click **New Product**.
2. In the **Name** field, enter a name to identify the product. This name populates the **Label** field.
3. In the **GPG Key** field, enter a GPG Key for the product.
4. From the **Sync Plan** list, select a synchronization plan for the product.
5. In the **Description** field, enter a description of the product.
6. Click **Save**.
7. In the Products window, click the new product and then click **Create Repository**.
8. In the **Name** field, enter a name for the repository. This automatically populates the **Label** field.
9. From the **Type** list, select **file**.
10. In the **Upstream URL** field, enter the URL of the registry to use as a source. Add a corresponding user name and password in the **Upstream Username** and **Upstream Password** fields.
11. Click **Save**.
12. Click the new repository.
13. Navigate to the **Upload File** and click **Browse**.
14. Select the **.iso** file and click **Upload**.

## For CLI Users

1. Create the custom product:

```
# hammer product create \  
--name "My_ISOs" \  
--sync-plan "Example Plan" \  
--description "My_Product" \  
--organization "My_Organization"
```

2. Create the repository:

```
# hammer repository create \  
--name "My_ISOs" \  
--content-type "file" \  
--product "My_Product" \  
--organization "My_Organization"
```

3. Upload the ISO file to the repository:

```
# hammer repository upload-content \  
--path ~/bootdisk.iso \  
--name "My_ISOs" \  
--organization "My_Organization"
```

## CHAPTER 13. MANAGING CUSTOM FILE TYPE CONTENT

In Satellite, you might require methods of managing and distributing SSH keys and source code files or larger files such as virtual machine images and ISO files. To achieve this, custom products in Red Hat Satellite include repositories for custom file types. This provides a generic method to incorporate arbitrary files in a product.

You can upload files to the repository and synchronize files from an upstream Satellite Server. When you add files to a custom file type repository, you can use the normal Satellite management functions such as adding a specific version to a Content View to provide version control and making the repository of files available on various Capsule Servers. Clients must download the files over HTTP or HTTPS using **curl -O**.

You can create a file type repository in Satellite Server only in a custom product, but there is flexibility in how you create the file type repository. You can create an independent file type repository in a directory on the system where Satellite is installed, or on a remote HTTP server, and then synchronize the contents of that directory into Satellite. This method is useful when you have multiple files to add to a Satellite repository.

### 13.1. CREATING A CUSTOM FILE TYPE REPOSITORY IN RED HAT SATELLITE

The procedure for creating a custom file type repository is the same as the procedure for creating any custom content, except that when you create the repository, you select the **file** type. You must create a product and then add a custom repository.

#### Procedure

To create a custom product, complete the following procedure:

1. In the Satellite web UI, navigate to **Content > Products**, click **Create Product** and enter the following details:
2. In the **Name** field, enter a name for the product. Red Hat Satellite 6 automatically completes the **Label** field based on what you have entered for **Name**.
3. Optional: From the **GPG Key** list, select the GPG key for the product.
4. Optional: From the **Sync Plan** list, select a synchronization plan for the product.
5. In the **Description** field, enter a description of the product, and then click **Save**.

To create a repository for your custom product, complete the following procedure:

1. In the Products window, select the name of a product that you want to create a repository for.
2. Click the **Repositories** tab, and then click **New Repository**.
3. In the **Name** field, enter a name for the repository. Red Hat Satellite 6 automatically completes the **Label** field based on the name.
4. From the **Type** list, select **file**.
5. In the **Upstream URL** field, enter the URL of the upstream repository to use as a source.

6. Select the **Verify SSL** check box if you want to verify that the upstream repository's SSL certificates are signed by a trusted CA.
7. In the **Upstream Username** field, enter the user name for the upstream repository if required for authentication. Clear this field if the repository does not require authentication.
8. In the **Upstream Password** field, enter the corresponding password for the upstream repository. Clear this field if the repository does not require authentication.
9. Click **Save**.

### For CLI Users

1. Create a Custom Product

```
# hammer product create \
--name "My File Product" \
--sync-plan "Example Plan" \
--description "My files" \
--organization "My_Organization"
```

**Table 13.1. Optional Parameters for the `hammer product create` Command**

Option	Description
<b>--gpg-key</b> <i>gpg_key_name</i>	Key name to search by
<b>--gpg-key-id</b> <i>gpg_key_id</i>	GPG key numeric identifier
<b>--sync-plan</b> <i>sync_plan_name</i>	Sync plan name to search by
<b>--sync-plan-id</b> <i>sync_plan_id</i>	Sync plan numeric identifier

2. Create a File Type Repository

```
# hammer repository create \
--name "My Files" \
--content-type "file" \
--product "My File Product" \
--organization "My_Organization"
```

**Table 13.2. Optional Parameters for the `hammer repository create` Command**

Option	Description
<b>--checksum-type</b> <i>sha_version</i>	Repository checksum, currently 'sha1' & 'sha256' are supported
<b>--download-policy</b> <i>policy_name</i>	Download policy for yum repos (either 'immediate', 'on_demand', or 'background').



Option	Description
<b>--gpg-key</b> <i>gpg_key_name</i>	Key name to search by
<b>--gpg-key-id</b> <i>gpg_key_id</i>	GPG key numeric identifier
<b>--mirror-on-sync</b> <i>boolean</i>	Must this repo be mirrored from the source, and stale RPMs removed, when synced? Set to <b>true</b> or <b>false</b> , <b>yes</b> or <b>no</b> , <b>1</b> or <b>0</b> .
<b>--publish-via-http</b> <i>boolean</i>	Must this also be published using HTTP? Set to <b>true</b> or <b>false</b> , <b>yes</b> or <b>no</b> , <b>1</b> or <b>0</b> .
<b>--upstream-username</b> <i>repository_username</i>	Upstream repository user, if required for authentication
<b>--upstream-password</b> <i>repository_password</i>	Password for the upstream repository user
<b>--url</b> <i>source_repo_url</i>	URL of the Source repository
<b>--verify-ssl-on-sync</b> <i>boolean</i>	Must Katello verify that the upstream URL's SSL certificates are signed by a trusted CA? Set to <b>true</b> or <b>false</b> , <b>yes</b> or <b>no</b> , <b>1</b> or <b>0</b> .

## 13.2. CREATING A CUSTOM FILE TYPE REPOSITORY IN A LOCAL DIRECTORY

You can create a custom file type repository, from a directory of files, on the base system where Satellite is installed using the **pulp-manifest** command. You can then synchronize the files into Satellite Server. When you add files to a file type repository, you can work with the files as with any other repository.

Use this procedure to configure a repository in a directory on the base system where Satellite is installed. To create a file type repository in a directory on a remote server, see [Section 13.3, "Creating a Remote File Type Repository"](#).

### Procedure

To create a file type repository in a local directory, complete the following procedure:

1. Ensure the Server and Satellite Tools repositories are enabled:

```
# subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-7-server-satellite-tools-6.4-rpms
```

2. Install the Pulp Manifest package:

```
# yum install python-pulp-manifest
```

3. Create a directory that you want to use as the file type repository in the HTTP server's public folder:

```
# mkdir my_file_repo
```

4. Add files to the directory or create a test file:

```
# touch my_file_repo/test.txt
```

5. Enter the Pulp Manifest command to create the manifest:

```
# pulp-manifest my_file_repo
```

6. Verify the manifest was created:

```
# ls my_file_repo  
PULP_MANIFEST test.txt
```

## Importing Files from a File Type Repository

To import files from a file type repository in a local directory, complete the following procedure:

1. Ensure a custom product exists in Satellite Server.
2. In the Satellite web UI, navigate to **Content > Products**.
3. Select the name of a product.
4. Click the **Repositories** tab and select **New Repository**.
5. In the **Name** field, enter a name for the repository. Red Hat Satellite 6 automatically completes this field based on what you enter for **Name**.
6. From the **Type** list, select the content type of the repository.
7. In the **Upstream URL** field, enter the local directory with the repository to use as the source, in the form **file:///my\_file\_repo**.
8. Select the **Verify SSL** check box to check the SSL certificate for the repository or clear the **Verify SSL** check box.
9. Optional: In the **Upstream Username** field, enter the upstream user name that you require.
10. Optional: In the **Upstream Password** field, enter the corresponding password for your upstream user name.
11. Select **Save** to save this repository entry.

## Updating a File Type Repository

To update the file type repository, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Products**.
2. Select the name of a product.
3. Select the name of the repository you want to update.
4. From the **Select Action** menu, select **Sync Now**.

5. Visit the URL where the repository is published to see the files.

### 13.3. CREATING A REMOTE FILE TYPE REPOSITORY

You can create a custom file type repository from a directory of files that is external to Satellite Server using the **pulp-manifest** command. You can then synchronize the files into Satellite Server over HTTP or HTTPS. When you add files to a file type repository, you can work with the files as with any other repository.

Use this procedure to configure a repository in a directory on a remote server. To create a file type repository in a directory on the base system where Satellite Server is installed, see [Section 13.2, "Creating a Custom File Type Repository in a Local Directory"](#).

#### Prerequisites

Before you create a remote file type repository, ensure the following conditions exist:

- You have a Red Hat Enterprise Linux 7 server registered to your Satellite or the Red Hat CDN.
- Your server has an entitlement to the Red Hat Enterprise Linux Server and Satellite Tools repositories.
- You have installed an HTTP server. For more information about configuring a web server, see [The Apache HTTP Server](#) in the Red Hat Enterprise Linux 7 *System Administrator's Guide*.

#### Procedure

To create a file type repository in a remote directory, complete the following procedure:

1. On your remote server, ensure that the Server and Satellite Tools repositories are enabled:

```
# subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-7-server-satellite-tools-6.4-rpms
```

2. Install the Pulp Manifest package:

```
# yum install python-pulp-manifest
```

3. Create a directory that you want to use as the file type repository in the HTTP server's public folder:

```
# mkdir /var/www/html/pub/my_file_repo
```

4. Add files to the directory or create a test file:

```
# touch /var/www/html/pub/my_file_repo/test.txt
```

5. Enter the Pulp Manifest command to create the manifest:

```
# pulp-manifest /var/www/html/pub/my_file_repo
```

6. Verify the manifest was created:

```
# ls /var/www/html/pub/my_file_repo  
PULP_MANIFEST test.txt
```

## Importing Files from a Remote a File Type Repository

To import files from a remote file type repository, complete the following procedure:

1. Ensure a custom product exists in Satellite Server, or create a custom product. For more information see [Section 13.1, "Creating a Custom File Type Repository in Red Hat Satellite"](#)
2. In the Satellite web UI, navigate to **Content > Products**.
3. Select the name of a product.
4. Click the **Repositories** tab and select **New Repository**.
5. In the **Name** field, enter a name for the repository. Red Hat Satellite 6 automatically completes this field based on what you enter for **Name**.
6. From the **Type** list, select **file**.
7. In the **Upstream URL** field, enter the URL of the upstream repository to use as a source.
8. Select the **Verify SSL** check box if you want to verify that the upstream repository's SSL certificates are signed by a trusted CA.
9. In the **Upstream Username** field, enter the user name for the upstream repository if required for authentication. Clear this field if the repository does not require authentication.
10. In the **Upstream Password** field, enter the corresponding password for the upstream repository. Clear this field if the repository does not require authentication.
11. Click **Save**.
12. To update the file type repository, navigate to **Content > Products**. Select the name of a product that contains the repository that you want to update.
13. In the product's window, select the name of the repository you want to update.
14. From the **Select Action** menu, select **Sync Now**.

Visit the URL where the repository is published to view the files.

## 13.4. UPLOADING FILES TO A CUSTOM FILE TYPE REPOSITORY IN RED HAT SATELLITE

### Procedure

To upload files to a custom file type repository, complete the following steps:

1. In the Satellite web UI, navigate to **Content > Products**.
2. Select a custom product by name.
3. Select a file type repository by name.
4. Click **Browse** to search and select the file you want to upload.

5. Click **Upload** to upload the selected file to Satellite Server.
6. Visit the URL where the repository is published to see the file.

### For CLI Users

```
# hammer repository upload-content \
--name "My Files" \
--organization "My_Organization" \
--path example_file
```

The **--path** option can indicate a file, a directory of files, or a glob expression of files. Globs must be escaped by single or double quotes.

## 13.5. DOWNLOADING FILES TO A HOST FROM A CUSTOM FILE TYPE REPOSITORY IN RED HAT SATELLITE

You can download files to a client over HTTPS using **curl -O**, and optionally over HTTP if the **Publish via HTTP** repository option is selected.

### Prerequisites

- You have a custom file type repository. See [Section 13.1, "Creating a Custom File Type Repository in Red Hat Satellite"](#) for more information.
- You know the name of the file you want to download to clients from the file type repository.
- To use HTTPS you require the following certificates on the client:
  1. The **katello-server-ca.crt**. For more information, see [Installing the Katello Root CA Certificate](#) in the *Administering Red Hat Satellite* guide.
  2. An Organization Debug Certificate. See [Section 2.3, "Creating an Organization Debug Certificate"](#) for more information.

### Procedure

To download files to a host from a custom file type repository, complete the following procedure:

1. In the Satellite web UI, navigate to **Content > Products**.
2. Select a custom product by name.
3. Select a file type repository by name.
4. Check to see if **Publish via HTTP** is enabled. If it is not, you require the certificates to use HTTPS.
5. Copy the URL where the repository is published.

### For CLI Users

1. List the file type repositories.

```
# hammer repository list --content-type file
```

```

---|-----|-----|-----|
ID | NAME   | PRODUCT   | CONTENT TYPE | URL
---|-----|-----|-----|
7  | My Files | My File Product | file         |
---|-----|-----|-----|

```

2. Display the repository information.

```
# hammer repository info --name "My Files" --product "My File Product" --organization-id 1
```

If HTTP is enabled, the output is similar to this:

```

Publish Via HTTP: yes
Published At:      http://satellite.example.com/pulp/isos/uuid/

```

If HTTP is not enabled, the output is similar to this:

```

Publish Via HTTP: no
Published At:      https://satellite.example.com/pulp/isos/uuid/

```

3. On the client, enter a command in the appropriate format for HTTP or HTTPS:

For HTTP:

```
# curl -O satellite.example.com/pulp/isos/uuid/my_file
```

For HTTPS:

```
# curl -O --cert ./Default\ Organization-key-cert.pem --cacert katello-server-ca.crt
satellite.example.com/pulp/isos/uuid/my_file
```

## CHAPTER 14. MANAGING CUSTOM PUPPET CONTENT

In Satellite, if you want to incorporate state configuration of hosts using Puppet modules, you can create a custom product with repositories for Puppet modules to achieve this.

### 14.1. CREATING A CUSTOM PUPPET REPOSITORY

The procedure for creating a custom Puppet module repository is the same as the procedure for creating any custom content, except that when you create the repository, you select the **puppet** type. You must create a product and then add a custom repository.

#### Procedure

1. In the Satellite web UI, navigate to **Content > Products**, and click the product that you want to use.
2. Click **Create Repository**.
3. In the **Name** field, enter a name for the repository. Red Hat Satellite 6 automatically completes the **Label** field based on what you have entered for **Name**.
4. From the **Type** list, select **puppet**.
5. In the **URL** field, enter the URL of the external repository to use as a source. You can use a repository source to synchronize your own Puppet modules.
6. Click **Save**.

#### For CLI Users

1. Enter the following command to create a Puppet module repository:

```
# hammer repository create \
--name "PostgreSQL Puppet Modules" \
--content-type "puppet" \
--product "PostgreSQL" \
--organization "My_Organization"
```

### 14.2. MANAGING INDIVIDUAL PUPPET MODULES

If you want to create a custom product that contains both RPM content and a Puppet module to install and configure a server using the custom RPM content, use the procedure in [Section 14.1, "Creating a Custom Puppet Repository"](#) and then use the following procedure to upload Puppet modules.

#### Support for Custom RPMs

Red Hat does not support the modules from Puppet Forge. For any issues with these modules, contact the module developer.

#### Prerequisites

1. From the Puppet Forge website, download the module that you want to use, for example, <https://forge.puppetlabs.com/puppetlabs/postgresql>.

2. In your web browser, click **download latest tar.gz** to save to your local file system.

### Procedure

1. In the Satellite web UI, navigate to **Content > Products** and select the product that contains the Puppet repository that you want to manage.
2. In the repository window, click the new Puppet repository, which displays the details page for that repository.
3. Navigate to the **Upload Puppet Module** area, click **Browse**, select the newly downloaded and extracted Puppet module, and click **Upload**.

To manage and remove Puppet modules from a product, complete the following steps:

1. In the window for your Puppet Modules repository, navigate to the upper right of the window to the **Content Counts** area. In the **Puppet Modules** row, click the numerical value that is displayed for the Puppet Modules.
2. In the **Manage Puppet Modules** for your Puppet Module repository window, select the modules that you want to manage and then click **Select Action** and perform an action, or select **Remove Puppet Modules**.

### For CLI Users

1. Copy the Puppet module to your Satellite Server's file system:

```
$ scp ~/puppet_module.tar.gz root@satellite.example.com:~/.
```

2. Import the Puppet module to the Puppet Modules repository:

```
# hammer repository upload-content \  
--path ~/puppet_module.tar.gz \  
--name "My Puppet Modules" \  
--organization "My_Organization"
```

## 14.3. SYNCHRONIZING PUPPET REPOSITORIES

In addition to creating a repository of uploaded Puppet modules, Satellite Server can synchronize a complete Puppet module repository. In this example, Satellite Server synchronizes the entire Puppet Forge repository.

### Support for Custom RPMs

Red Hat does not support the modules from Puppet Forge. The modules are used to demonstrate the synchronization process. For any issues with these modules, contact the module developer.

### Procedure

1. In the Satellite web UI, navigate to **Content > Products** and click **New Product**.
2. In the **Name** field, enter a name for the product. Red Hat Satellite 6 automatically completes the **Label** field based on what you have entered for **Name**.
3. Optional: From the **GPG Key** list, select the GPG key for the product.



4. Optional: From the **Sync Plan** list, select a synchronization plan for the product.
5. In the **Description** field, enter a description of the product.
6. Click **Save**.
7. Click **Create Repository**, which displays a form for a new repository.
8. In the **Name** field, enter a name for the repository. Red Hat Satellite 6 automatically completes this field based on what you have entered for **Name**.
9. From the **Type** list, select **puppet**.
10. In the **URL** field, enter <http://forge.puppetlabs.com/>.
11. Click **Save**.
12. Select the new Puppet repository and click **Sync Now** to import all modules from Puppet Forge into Satellite Server. This can take a long time.

### For CLI Users

1. Create the product:

```
# hammer product create \
--name "Puppet Forge" \
--sync-plan "Example Plan" \
--description "All modules from Puppet Forge" \
--organization "My_Organization"
```

2. Create the Puppet Forge repository:

```
# hammer repository create \
--name "Puppet Forge Modules" \
--content-type "puppet" \
--product "Puppet Forge" \
--organization "My_Organization" \
--url http://forge.puppetlabs.com/
```

3. Synchronize the repository:

```
# hammer repository synchronize \
--name "Puppet Forge Modules" \
--product "Puppet Forge" \
--organization "My_Organization"
```

The Puppet Forge repository contains several thousand modules and can take a long time to synchronize.

## 14.4. SYNCHRONIZING PUPPET MODULES FROM A GIT REPOSITORY

Red Hat Satellite 6 includes a utility called **pulp-puppet-module-builder**, which you can install on other systems from the **pulp-puppet-tools** RPM. This tool checks out a Git repository, builds all the modules, and publishes them in a structure that Satellite 6 can synchronize. One common method is to run the

utility on Satellite Server itself, publish to a local directory, and synchronize against that directory. For example:

```
# mkdir /modules
# chmod 755 /modules
# pulp-puppet-module-builder \
--output-dir=/modules \
--url=git@mygitserver.com:mymodules.git \
--branch=develop
```

This example checks out the **develop** branch of the Git repository from **git@mygitserver.com:mymodules.git** and publishes it to **/modules**. Add this directory as the URL (**file:///modules**) for a new repository on Satellite Server.

## Publishing Puppet Modules on a Remote HTTP Server

The same process also applies to publishing modules on a remote HTTP server. For example, if you use **webserver.example.com** as a standard web host to publish the Puppet modules.

```
# mkdir /var/www/html/modules/
# chmod 755 /var/www/html/modules/
# pulp-puppet-module-builder \
--output-dir=/var/www/html/modules/ \
--url=git@mygitserver.com:mymodules.git \
--branch=develop
```

On Satellite Server, set the repository's URL to <http://webserver.example.com/modules/>.

## Synchronizing Puppet Modules from a Git repository using the web UI

Use the following procedure to synchronize Puppet modules from a Git repository.

### Procedure

1. Create a custom product and click **Create Repository**.
2. From the **Type** list, select **puppet**.
3. In the **URL** field, enter the URL of the external Git repository to use as a source in the following format: **file:///modules**.

### For CLI Users

1. Create the Puppet Forge repository:

```
# hammer repository create \
--name "Modules from Git" \
--content-type "puppet" \
--product "MyProduct" \
--organization "My_Organization" \
--url file:///modules
```

## CHAPTER 15. SYNCHRONIZING CONTENT BETWEEN SATELLITE SERVERS

Red Hat Satellite 6.4 uses Inter-Satellite Synchronization (ISS) to synchronize content between upstream and downstream servers. In the context of ISS, upstream refers to the server from which content is exported; downstream refers to the server into which content is imported.

ISS is designed to address two scenarios:

- If you have both connected and disconnected Satellite Servers, and want to copy content from the connected servers to the disconnected servers. The only way that these disconnected Satellites are populated with content is by exports from a connected Satellite.
- If you have a primary Satellite Server and want to promote some, but not all, content to other Satellite Servers. For example, you might have Content Views (CVs) that are validated by the IT department, and you want to promote the **yum** content from those CVs to a downstream Satellite.

If you do not want to maintain another management web UI and platform, and you want to perform management and provisioning to local clients, consider setting up a Capsule Server.

Bidirectional synchronization is not used in a disconnected environment. Content never passes from the disconnected server to the connected server.

You cannot synchronize content from Satellite Server to Capsule Server. Capsule Server supports synchronization natively.

### Supported Synchronization Options

Satellite 6.4 supports the following synchronization options:

- Exporting Content View versions to a directory or ISO file. For more information, see [Section 15.2, “Synchronizing Content Using Content View Versions”](#).
- Exporting repositories to a directory or ISO file. For more information, see [Section 15.3, “Synchronizing Content Using Repositories”](#).

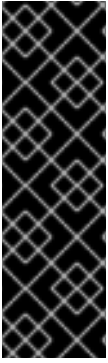
Only RPM, kickstart, and ISO files are exported. Content View definitions and metadata, such as package filters, is not transferred. Satellite does not support the export of Puppet, Docker, or OSTree content. Imports occur as a normal repository synchronization, and consequently always arrive in the Library environment.

These synchronization options include a range of history details about the export and import, depending on the type of content. For example:

- The repository synchronization history includes upstream source information, as well as the time the export occurred.
- The CV synchronization history includes the export time and version, as well as the import time, version, and upstream source.

### 15.1. CONFIGURING AN EXPORT DESTINATION

Inter-Satellite Synchronization by default uses the `/var/lib/pulp/katello-export/` directory, as indicated by the `pulp_export_destination` setting. To change this directory you must create the new directory and configure the Pulp export destination setting.



## IMPORTANT

The directory used in this example is for demonstration purposes only. Confirm that the export directory has adequate space for the required export RPM and ISO files. [Section 1.4, "Content Management Storage"](#) provides information on estimating storage requirements. A temporary file is created during the export process in the `/var/cache/pulp/` directory, then transferred to the `/var/lib/pulp/` directory. This means that storage space equal to twice the size of the repository being exported is required during the export process. The temporary file is deleted when the export is completed.

### Procedure

1. Create the export directory:

```
# mkdir /var/www/html/pub/export
```

2. Ensure the **foreman** user has read and write permissions on the export directory:

```
# chown foreman:foreman /var/www/html/pub/export
```

3. Configure the SELinux context:

```
# semanage fcontext -a -t httpd_sys_rw_content_t \
"/var/www/html/pub/export(/.*)?"
# restorecon -RvF /var/www/html/pub/export
# ls -Zd /var/www/html/pub/export
drwxr-xr-x. foreman foreman system_u:object_r:httpd_sys_rw_content_t:s0
/var/www/html/pub/export
```

4. Change the export destination:

```
# hammer settings set \
--name pulp_export_destination \
--value your-export-directory
```

## 15.2. SYNCHRONIZING CONTENT USING CONTENT VIEW VERSIONS

You can use Content Views with multiple repositories to synchronize content between Satellite Servers. To do that, you must export a Content View version on Satellite Server, copy the exported directory or ISO files to another Satellite Server and import them there.

### 15.2.1. Creating a Content View for Exporting

Use this procedure to create a Content view and prepare it for exporting. You must create filters to include non-errata and errata products per date.

#### Procedure

1. Navigate to **Content > Content Views**. Click **Create New View**.
2. In the **Name** field, enter **Export\_CV**.

3. Click **Save**.
4. In the **Repository Selection** area, select the repositories to be added to the CV and click **Add Repositories**.
5. Navigate to **Yum Content > Filters** and click **New Filter**.
6. In the **Name** field, enter **Non-errata Products**.
7. From the **Content type** list, select **Package**.
8. From the **Inclusion type** list, select **Include**.
9. In the **Description** field, enter **Include all non-errata Products**.
10. Click **Save**.
11. Select the **Include all RPMs with no errata** check box.
12. Navigate to **Yum Content > Filters** and click **New Filter**.
13. In the **Name** field, enter **Erratas until YYYY-MM-DD**.
14. From the **Content type** list, select **Erratum - Date and Type**.
15. From the **Inclusion type** list, select **Include**.
16. In the **Description** field, enter **Include errata products until YYYY-MM-DD**.
17. Click **Save**.
18. In the **Errata Type** area, select all **Security, Enhancement** and **Bugfix** errata types.
19. Select the **Updated On** check box in **Data type**.
20. Fill in the **Start Date** and **End Date** fields to configure the date range of Products for the filter.
21. Click **Save**.
22. Click **Publish New Version**, fill in the date ranges of repositories into the **Description** field. Clear the **Force Yum Metadata Regeneration** check box.
23. Click **Save** to publish the CV version ready for exporting.

### For CLI Users

1. Enter the **hammer content-view create** command to create a new CV:

```
# hammer content-view create \
--name "Export_CV" \
--organization "Default Organization"
```

2. Enter the **hammer content-view add-repository** command to add repositories to the CV:

```
# hammer content-view add-repository \
--name "Export_CV" \
--product "Red Hat Satellite" \
```

```
--repository "Red Hat Satellite Tools 6 for RHEL 7 Server RPMs x86_64" \
--organization "Default Organization"
# hammer content-view add-repository \
--name "Export_CV" \
--product "Red Hat Satellite Capsule" \
--repository "Red Hat Satellite Capsule Tools 6.2 for RHEL 7 Server RPMs x86_64" \
--organization "Default Organization"
```

3. Enter the **hammer content-view filter create** command to create a filter for including non-errata packages:

```
# hammer content-view filter create \
--content-view "Export_CV" \
--inclusion true \
--name "Non-errata_Products" \
--type rpm \
--original-packages true \
--organization "Default Organization"
```

4. Enter the **hammer content-view filter create** command to create a filter for including errata packages:

```
# hammer content-view filter create \
--content-view "Export_CV" \
--inclusion true \
--name "Erratas until YYYY-MM-DD" \
--type erratum \
--organization "Default Organization"
```

5. Enter the **hammer content-view filter rule create** command to create a rule defining the date range:

```
# hammer content-view filter rule create \
--content-view "Export_CV" \
--content-view-filter "Erratas until YYYY-MM-DD" \
--end-date YYYY-MM-DD \
--types security,enhancement,bugfix \
--organization "Default Organization"
```

6. Enter the **hammer content-view publish** command to publish the CV version ready for exporting. It is recommended to fill in the date ranges of repositories under the **--description** option.

```
# hammer content-view publish \
--name "Export_CV" \
--description "Repositories until YYYY-MM-DD" \
--force-yum-metadata-regeneration true \
--async \
--organization "Default Organization"
```

### 15.2.2. Exporting a Content View Version

You can export a version of a Content View to a directory. That means that you can label a particular version of a CV to suit your requirements. This way you can curate and track your exports and facilitate updates.

Satellite exports only RPM, kickstart, and ISO files. Content View definitions and metadata, such as package filters, are not exported. Satellite 6.4 does not support the export of Puppet, Docker, or OSTree content.

## Using Chunked ISO Files

Satellite 6.4 supports exports to chunked ISO files. A chunked ISO is similar to a split ISO but with one significant difference. Satellite tracks the size of the ISO file, and if the total size of the files being added to the ISO exceeds that value, Satellite stops writing to the ISO and creates a new one in the series. The advantage of this is that you can specify the ISO file size (for example, 4.7 GB), and still export larger repositories. The result is multiple 4.7 GB ISO files that you can burn to DVDs.

You can add the **--iso-mb-size** parameter to specify the size of ISO export files. The default value is 4380 MB, the size of a single-sided, single-layer DVD.

## Prerequisites

- Ensure that the export directory is large enough to accommodate at least one Red Hat Enterprise Linux export. By default, the export directory is **/var/lib/pulp/katello-export/**.
- Ensure that the **/var/lib/pulp/** directory has free storage space equivalent to the size of the repositories being exported for temporary files created during the export process.
- Ensure that the **/var/cache/pulp** directory has free storage space equivalent to twice of the size of the repository being exported for temporary files created during the export process.
- Ensure that you set download policy to **Immediate** for all repositories within the Content View you export. For more information, see [Section 5.4, "Configuring Download Policies"](#).
- Ensure that you synchronize repositories within the Content View you export to the required date.

## Procedure

Complete these steps to export a version of a Content View. Satellite exports versions of Content Views to a directory by default. You can add the **--export-to-iso 1** parameter to export to an ISO file instead.

1. Enter the **hammer content-view version list** command to determine which version of a Content View to export:

```
$ hammer content-view version list --organization "Default Organization"
```

2. Enter the **hammer content-view version export** command to export a version of a Content View. This command exports content to the directory specified in **pulp\_export\_destination** setting. For more information, see [Section 15.1, "Configuring an Export Destination"](#).

```
# hammer content-view version export --id 3
```

## Incremental Export of a Content View Version

To avoid exporting large repositories when updating, you can use incremental updates to only export the recent updates from a Satellite Server. Incremental updates export the recent changes made in the local repository by using synchronization events set to a particular date and time.

The time stamp used for the calculations is the time that the RPMs were synchronized on Satellite Server. For example, if Red Hat adds RPMs to a repository on a Monday and then again on a Wednesday, you can synchronize your local repository on Thursday and then use the date of Wednesday to get only the Wednesday update.

To make an incremental-update repository, enter the **hammer content-view version export** command with the **--since** option. For example, the following command exports repositories that are 10 days old:

```
# hammer content-view version export --content-view-id 1 \
--since $(date +%Y-%m-%dT%H:%M:%S%Z' -d "now-10days")
```

### 15.2.3. Importing a Content View as a Red Hat Repository

Use this procedure to import content that you exported from a Content View from Satellite Server. You can also recreate any custom products during the import process, but Red Hat products are not recreated because they must be created using a manifest.

#### Prerequisites

- Export the Content View from Satellite Server. For more details, see [Section 15.2.2, “Exporting a Content View Version”](#).
- Ensure that the Satellite Server where you want to import content has the required manifests and entitlements for any content that you want to enable. You cannot enable repositories for which no entitlements exist.

#### To Import a Content View:

1. Copy the exported directory to the **/var/www/html/pub/export/** directory on the Satellite Server where you want to import content.
2. In the web UI, navigate to **Content > Subscriptions**.
3. Select **Manage Manifests**.
4. On the **Import/Remove Manifest** tab, set the **Red Hat CDN URL** address field to match the location of a **content** directory and a **listing** file within the exported Content View. For example, if the exported CV is located in **/var/www/html/pub/export/Default\_Organization-Export\_CV-v1.0**, set the URL to the following form:

```
http://satellite.example.com/pub/export/Default_Organization-Export_CV-
v1.0/Default_Organization/content_views/Export_CV/1.0/
```

5. Click **Save**.
6. Navigate to **Content > Red Hat Repositories** and enable the exported repository.
7. On the Satellite Server where you import content, enter the **hammer organization update** command to add new repositories to an organization. Set the **redhat-repository-url** option to the directory corresponding to the version you want within the exported Content View.

```
$ hammer organization update \
--name "Default Organization" \
```



```
--redhat-repository-url \
http://satellite.example.com/pub/export/Default_Organization-Export_CV-
v1.0/Default_Organization/content_views/Export_CV/1.0/
```

Organization updated

## 15.3. SYNCHRONIZING CONTENT USING REPOSITORIES

You can synchronize single repositories between Satellite Servers. To do that, you must export a repository on Satellite Server, copy the exported directory or ISO file to another Satellite Server and import it there.

### 15.3.1. Exporting a repository

Satellite exports repositories to a directory by default. You can add the **--export-to-iso 1** parameter to export to an ISO file instead.

Satellite only exports RPM, kickstart, and ISO files. Satellite does not support the export of Puppet, Docker, or OSTree content. Imports occur as a normal repository synchronization, and consequently always arrive in the Library environment.

#### Using Chunked ISO Files

Satellite 6.4 supports exports to chunked ISO files. A chunked ISO is similar to a split ISO but with one significant difference. Satellite tracks the size of the ISO file, and if the total size of the files being added to the ISO exceeds that value, Satellite stops writing to the ISO and creates a new one in the series. The advantage of this is that you can specify the ISO file size (for example, 4.7 GB), and still export larger repositories. The result is multiple 4.7 GB ISO files that you can burn to DVDs.

You can add the **--iso-mb-size** parameter to specify the size of ISO export files. The default value is 4380 MB, the size of a single-sided, single-layer DVD.

#### Prerequisites

- Ensure that the export directory is large enough to accommodate at least one Red Hat Enterprise Linux export. By default, the export directory is **/var/lib/pulp/katello-export/**.
- Ensure that the **/var/lib/pulp/** directory has free storage space equivalent to the size of the repositories being exported for temporary files created during the export process.
- Ensure that the **/var/cache/pulp** directory has free storage space equivalent to twice of the size of the repository being exported for temporary files created during the export process.
- Ensure that you set download policy to **Immediate** for all repositories you export. For more information, see [Section 5.4, "Configuring Download Policies"](#).
- Ensure that you synchronize repositories you export to the required date.

#### Procedure

1. Enter the **hammer repository list** command to list the repositories which can be exported and identify their IDs, which are used in the export.
2. Enter the **hammer repository export** command to export content from your Satellite Server. This command exports content to the directory specified in **pulp\_export\_destination** setting.

—

```
# hammer repository export --id 1
```

### 15.3.2. Incremental Export of a Repository

To avoid exporting large repositories when updating, you can use incremental updates to only export the recent updates from a Satellite Server. Incremental updates export the recent changes made in the local repository by using synchronization events set to a particular date and time.

The time stamp used for the calculations is the time that the RPMs were synchronized on Satellite Server. For example, if Red Hat adds RPMs to a repository on a Monday and then again on a Wednesday, you can synchronize your local repository on Thursday and then use the date of Wednesday to get only the Wednesday update.

To make an incremental-update repository, enter the **hammer repository export** command with the **--since** option. For example:

```
# hammer repository export \  
--id 1 \  
--since 2010-01-01T12:00:00Z
```

### 15.3.3. Importing a Repository

Use this procedure to import repository to Satellite Server using the exported directory or ISO file. You can also recreate any custom products during the import process, but Red Hat products are not recreated because they must be created using a manifest.

#### Prerequisites

- Export the repository from Satellite Server. For more details, see [Section 15.3.1, “Exporting a repository”](#).
- Ensure that the Satellite Server where you want to import content has the required manifests and entitlements for any content that you want to enable. You cannot enable repositories for which no entitlements exist.

#### Procedure

1. Copy the exported directory to the **/var/www/html/pub/export/** directory on the Satellite Server where you want to import a repository.
2. In the web UI, navigate to **Content > Subscriptions**.
3. Select **Manage Manifests**.
4. On the **Import/Remove Manifest** tab, set the **Red Hat CDN URL** address field to match the location of a **content** directory and a **listing** file within the exported repository. For example, if the exported repository is located in **/var/www/html/pub/export/Default\_Organization-Red\_Hat\_Enterprise\_Linux\_7\_Server\_RPMs\_x68\_64**, set the URL to the following form:

```
http://satellite.example.com/pub/export/Default_Organization-  
Red_Hat_Enterprise_Linux_7_Server_RPMs_x68_64/Default_Organization/Library/
```

5. Click **Save**.

6. Navigate to **Content** > **Red Hat Repositories** and enable the exported repository.

### 15.3.4. Incremental Import of a Repository

When synchronizing from an incremental update, you must use the incremental synchronization settings. In the Satellite web UI, ensure the **Mirror on Sync** setting is disabled for the repositories before synchronizing.

If you are using the Hammer CLI, you must add the **--incremental** option. If you do not use the incremental synchronization settings, the repository content will be deleted and only the content from the incremental update will remain. Recovering from an accidental overwrite requires a full export and then synchronizing again.

## APPENDIX A. USING AN NFS SHARE FOR CONTENT STORAGE

Your environment requires adequate hard disk space to fulfill content storage. In some situations, it is useful to use an NFS share to store this content. This appendix shows how to mount the NFS share on your Satellite Server's content management component.



### IMPORTANT

Do not mount the full **/var/lib/pulp** on an NFS share. Use high-bandwidth, low-latency storage for the **/var/lib/pulp** file system. Red Hat Satellite has many I/O-intensive operations; therefore, high-latency, low-bandwidth storage might have issues with performance degradation. Only use the NFS share for the **/var/lib/pulp/content** directory.

1. Create the NFS share. This example uses a share at **nfs.example.com:/satellite/content**. Ensure this share provides the appropriate permissions to Satellite Server and its **apache** user.

2. Shutdown the Satellite services on the Satellite host:

```
# foreman-maintain service stop
```

3. Ensure Satellite Server has the **nfs-utils** package installed:

```
# yum install nfs-utils
```

4. You need to copy the existing contents of **/var/lib/pulp/content** to the NFS share. First, mount the NFS share to a temporary location:

```
# mkdir /mnt/temp
# mount -o rw nfs.example.com:/satellite/content /mnt/temp
```

Copy the existing contents of **/var/lib/pulp/content** to the temporary location:

```
# cp -r /var/lib/pulp/content/* /mnt/temp/.
```

5. Set the permissions for all files on the share to use the **apache** user. This ID of this user is usually 48.
6. Unmount the temporary storage location:

```
# umount /mnt/temp
```

7. Remove the existing contents of **/var/lib/pulp/content**:

```
# rm -rf /var/lib/pulp/content/*
```

8. Edit the **/etc/fstab** file and add the following line:

```
nfs.example.com:/satellite/content /var/lib/pulp/content nfs
rw,hard,intr,context="system_u:object_r:httpd_sys_rw_content_t:s0"
```

This makes the mount persistent across system reboots. Ensure to include the SELinux context.

9. Enable the mount:

```
# mount -a
```

10. Confirm the NFS share mounts to **var/lib/pulp/content**:

```
# df
Filesystem              1K-blocks    Used Available Use% Mounted on
...
nfs.example.com:/satellite/content 309506048 58632800 235128224 20%
/var/lib/pulp/content
...
```

Also confirm that the existing content exists at the mount on **var/lib/pulp/content**:

```
# ls /var/lib/pulp/content
```

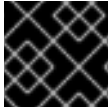
11. Start the Satellite services on the Satellite host:

```
# foreman-maintain service start
```

Satellite Server now uses the NFS share to store content. Run a content synchronization (see [Section 5.3, "Synchronizing Content"](#)) to ensure the NFS share works as expected.

## APPENDIX B. IMPORTING CONTENT ISOS INTO A DISCONNECTED SATELLITE

In high security environments where hosts are required to function in a closed network disconnected from the Internet, Satellite Server can provision systems with the latest security updates, errata, and packages. To accomplish this, download the Content ISOs for Red Hat Satellite from the Red Hat Customer Portal and import them into Satellite Server.



### IMPORTANT

This section is not required if your Satellite Server is connected to the Internet.

Download the product ISO from the Red Hat Customer Portal, as follows:

1. Navigate to Downloads (at the very top of the window) and select Red Hat Satellite.
2. Open the Content ISOs tab. All products in your subscription are listed here.
3. Click the link for the product name, such as Red Hat Enterprise Linux 6 Server (x86\_64) to download the ISO.
4. Copy all of Satellite Content ISOs to a directory Satellite can access. This example uses **/root/isos**.
5. Create a local directory that are shared through httpd on the Satellite. This example uses **/var/www/html/pub/sat-import/**.

```
# mkdir -p /var/www/html/pub/sat-import/
```

6. Mount and recursively copy the contents of the first ISO to the local directory:

```
# mkdir /mnt/iso
# mount -o loop /root/isos/first_iso /mnt/iso
# cp -ruv /mnt/iso/* /var/www/html/pub/sat-import/
# umount /mnt/iso
# rmdir /mnt/iso
```

7. Repeat the above step for each ISO until you have copied all the data from the Content ISOs into **/var/www/html/pub/sat-import/**.
8. Ensure the SELinux contexts for the directory are correct:

```
# restorecon -rv /var/www/html/pub/sat-import/
```

9. Satellite Server now contains the content from the Content ISOs. However, Satellite Server needs to point to this location as the CDN URL. In the Satellite Web UI, navigate to **Content > Red Hat Subscriptions**.
10. Click **Manage Manifest**.
11. On the Subscription Manifest information screen, select the **Actions** tab.
12. Navigate to Red Hat Provider Details. Click the edit icon on the **Red Hat CDN URL** and change the URL to the Satellite host name with the newly created directory, for example:

<http://server.example.com/pub/sat-import/>

13. Click **Save** and then upload your manifest using [Section 4.1.8, "Importing a Subscription Manifest into Satellite Server"](#).

Satellite is now acting as its own CDN with the files located in <http://server.example.com/pub/sat-import/>. This is not a requirement. The CDN can be hosted on a different machine inside the same disconnected network as long as it is accessible to Satellite Server using HTTP.

If your environment changes from disconnected to connected, you can reconfigure a disconnected Satellite to pull content directly from Red Hat Customer Portal:

1. In the Satellite Web UI, navigate to **Content > Red Hat Subscriptions**
2. Click **Manage Manifest**.
3. On the Subscription Manifest information screen, select the **Actions** tab.
4. Navigate to Red Hat Provider Details. Click the edit icon on the **Red Hat CDN URL** and change the URL to the Red Hat CDN URL:  
<https://cdn.redhat.com>
5. Click **Save**

Satellite Server pulls content directly from Red Hat Customer Portal on the next synchronization.

## APPENDIX C. IMPORTING CONTENT ISOS INTO A CONNECTED SATELLITE

Even if Satellite Server can connect directly to the Red Hat Customer Portal, you can perform the initial synchronization from locally mounted content ISOs. When the initial synchronization is completed from the content ISOs, you can switch back to downloading content through the network connection. To accomplish this, download the Content ISOs for Red Hat Satellite from the Red Hat Customer Portal and import them into Satellite Server. For locations with bandwidth limitations, using an **On Demand** or **Background** download policy might be more efficient than downloading and importing Content ISOs.

Note that if you synchronize a Red Hat Enterprise Linux ISO, all minor versions of Red Hat Enterprise Linux also synchronize. You require adequate storage on your Satellite to account for this.



### IMPORTANT

This section is not required if your Satellite Server is connected to the Internet.

This example shows how to perform the first synchronization of the Red Hat Enterprise Linux 7 repository from content ISOs.

#### Downloading the content ISOs from the Red Hat Customer Portal

1. In your browser, navigate to [Red Hat Customer Portal](#) and log in.
2. Click **DOWNLOADS**.
3. Select **Red Hat Satellite**.
4. Select the **Content ISOs** tab.
5. Search for the section required, in this example **Red Hat Enterprise Linux 7**.
6. Click the link for the product name to reveal the ISO files.
7. Using your browser, download the required ISOs to a location accessible by your browser. For example, to your workstation's **Downloads** directory.

#### Importing the Content ISOs

1. In a terminal connected to Satellite Server, create a directory to act as a temporary store for all of the required Satellite Content ISOs. This example uses **/tmp/isos/rhel7**:

```
# mkdir -p /tmp/isos/rhel7
```

2. On your workstation, copy the ISO files to Satellite Server:

```
$ scp ~/Downloads/iso_file root@satellite.example.com:/tmp/isos/rhel7
```

3. On Satellite Server, create a directory to serve as a mount point for the ISOs:

```
# mkdir /mnt/iso
```

4. Create a working directory to hold the contents of all the ISOs:



```
# mkdir /mnt/rhel7
```

5. Mount and recursively copy the contents of the first ISO to the working directory:

```
# mount -o loop /tmp/isos/iso_file /mnt/iso
# cp -ruv /mnt/iso/* /mnt/rhel7/
# umount /mnt/iso
```

6. Repeat the above step for each ISO until you have copied all the data from the Content ISOs into **/mnt/rhel7**.
7. If required, remove the empty directory used as the mount point:

```
# rmdir /mnt/iso
```

8. If required, remove the temporary working directory and its contents to regain the space:

```
# rm -rf /tmp/isos/
```

### Performing the Initial Synchronization

1. Set the owner and the SELinux context for the directory and its contents to be the same as **/var/lib/pulp**:

```
# chcon -R --reference /var/lib/pulp /mnt/rhel7/
# chown -R apache:apache /mnt/rhel7/
```

2. Create or edit the **/etc/pulp/content/sources/conf.d/local.conf** file. Insert the following text into the file:

```
[rhel-7-server]
enabled: 1
priority: 0
expires: 3d
name: Red Hat Enterprise Linux 7 Server
type: yum
base_url: file:///mnt/rhel7/content/dist/rhel/server/7/7Server/x86_64/os/
```

The **base\_url** path might differ in your content ISO. The directory specified in **base\_url** must contain the **repodata** directory, otherwise the synchronization fails. To synchronize multiple repositories, create a separate entry for each of them in the configuration file **/etc/pulp/content/sources/conf.d/local.conf**.

3. In the Satellite web UI, navigate to **Content > Red Hat Repositories** and enable the following repository: **Red Hat Enterprise Linux 7 Server RPMs x86\_64 Server**
4. Under **Content > Sync Status** select the repository to be synchronized and click **Synchronize Now**.

Note that there is no indication in the Satellite web UI of which source is being used. In case of problems with a local source, Satellite pulls content through the network. To monitor the process, enter the following command in a terminal (limited to Red Hat Enterprise Linux 7 base systems):

```
# journalctl -f -l SYSLOG_IDENTIFIER=pulp | grep -v worker[^\,\.]heartbeat
```

The above command displays interactive logs. First, Satellite Server connects to the Red Hat Customer Portal to download and process repository metadata. Then, the local repository is loaded. In case of any errors, cancel the synchronization in the Satellite web UI and verify your configuration.

After successful synchronization you can detach the local source by removing its entry from **/etc/pulp/content/sources/conf.d/local.conf**.

## APPENDIX D. SYNCHRONIZING TEMPLATES WITH GIT

Red Hat Satellite 6 enables synchronization of Job Templates, Provisioning Templates, and Partition Table Templates between Satellite Server and a Git repository or a local directory.



### NOTE

Synchronizing templates between Satellite Server and a Git repository or a local directory is a Technology Preview feature. Technology Preview features are not fully supported under Red Hat Subscription Service Level Agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process.

This section details the workflow for:

- installing and configuring the TemplateSync plug-in
- performing exporting and importing tasks

### D.1. ENABLING THE TEMPLATESYNC PLUG-IN

1. To enable the plug-in on your Satellite Server:

```
# satellite-installer --enable-foreman-plugin-templates
```

2. To verify that the plug-in is installed correctly, ensure **Administer** > **Settings** includes the **TemplateSync** menu.

### D.2. CONFIGURING THE TEMPLATESYNC PLUG-IN

Navigate to **Administer** > **Settings** > **TemplateSync** to configure the plug-in. The following table explains the attributes behavior. Note that some attributes are only used on importing or exporting tasks.

Table D.1. Synchronizing Templates Plug-in configuration

Parameter	API parameter name	Meaning on importing	Meaning on exporting
Associate	<b>associate</b>  Accepted values: <b>always, new, never</b>	Associates templates with OS, Organization, and Location based on metadata.	N/A
Branch	<b>branch</b>	Specifies the default branch in Git repository to read from.	Specifies the default branch in Git repository to write to.
Dirname	<b>dirname</b>	Specifies the subdirectory under the repository to read from.	Specifies the subdirectory under the repository to write to.

Parameter	API parameter name	Meaning on importing	Meaning on exporting
Filter	<b>filter</b>	Imports only templates with names that match this regular expression.	Exports only templates with names that match this regular expression.
Force import	<b>force</b>	Imported templates overwrite locked templates with the same name.	N/A
Metadata export mode	<b>metadata_export_mode</b>  Accepted values: <b>refresh, keep, remove</b>	N/A	Defines how metadata is handled when exporting: <ul style="list-style-type: none"> <li>● <b>Refresh</b> – remove existing metadata from the template content and generate new metadata based on current assignments and attributes.</li> <li>● <b>Keep</b> – retain the existing metadata.</li> <li>● <b>Remove</b> – export template without metadata. Useful if you want to add metadata manually.</li> </ul>
Negate	<b>negate</b>  Accepted values: <b>true, false</b>	Imports templates ignoring the filter attribute.	Exports templates ignoring the filter attribute.
Prefix	<b>prefix</b>	Adds specified string to the beginning of the template if the template name does not start with the prefix already.	N/A
Repo	<b>repo</b>	Defines the path to the repository to synchronize from.	Defines the path to a repository to export to.

Parameter	API parameter name	Meaning on importing	Meaning on exporting
Verbosity	<b>verbose</b>  Accepted values: <b>true</b> , <b>false</b>	Enables writing verbose messages to the logs for this action.	N/A

## D.3. IMPORTING AND EXPORTING TEMPLATES

Importing and exporting tasks are available through a series of API calls. API calls use the role-based access control system, which enables the tasks to be executed as any user. The TemplateSync plug-in allows synchronizing with a Git repository or a local directory.

### Prerequisites

For imported templates to appear in the Satellite web UI, each template must contain the location and organization that the template belongs to. This applies to all template types. Before you import a template, ensure that you add the following section to the template:

```
<%#
kind: provision
name: My Kickstart File
oses:
- RedHat 7
- RedHat 6
locations:
- First Location
- Second Location
organizations:
- Default Organization
- Extra Organization
%>
```

You can also import and export templates using Hammer. For more information, see [Provisioning Templates](#) in the *Hammer Cli Guide*.

### D.3.1. Synchronizing Templates with a Git repository

1. Configure a Git server that uses SSH authorization, for example gitosis, gitolite, or git daemon.
2. Configure the TemplateSync plug-in settings on a **TemplateSync** tab.
  - a. Change the **Branch** setting to match the target branch on a Git server.
  - b. Change the **Repo** setting to match the Git repository. For example, for the repository located in **git@git.example.com/templates.git** set the setting into **ssh://git@git.example.com/templates.git**.
3. Accept Git SSH host key as the foreman user:

```
# sudo -u foreman ssh git.example.com
```

You can see the **Permission denied, please try again.** message in the output, which is expected, because the SSH connection cannot succeed yet.

4. Create an SSH key pair if you do not already have it. Do not specify any passphrase.

```
# sudo -u foreman ssh-keygen
```

5. Configure your Git server with the public key from your Satellite, which resides in `/usr/share/foreman/.ssh/id_rsa.pub`.
6. Export templates from your Satellite Server to the Git repository specified in the **TemplateSync** menu:

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://satellite.example.com/api/v2/templates/export \
-X POST

{"message":"Success"}
```

7. Import templates to Satellite Server after their content was changed:

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://satellite.example.com/api/v2/templates/import \
-X POST

{"message":"Success"}
```

Note that templates provided by Satellite are locked and you cannot import them by default. To overwrite this behavior, change the **Force import** setting in the **TemplateSync** menu to **yes** or add the **force** parameter `-d '{"force": "true"}'` to the import command.

### D.3.2. Synchronizing templates with a local directory

Synchronizing templates with a local directory is useful if you have configured any revision control system repository in the local directory. That way, you can edit templates and track the history of edits in the directory. You can also synchronize changes to Satellite Server after editing the templates.

1. Create the directory where templates are stored and apply appropriate permissions and SELinux context:

```
# mkdir -p /usr/share/templates_dir/
# chown foreman /usr/share/templates_dir/
# chcon -t httpd_sys_rw_content_t /usr/share/templates_dir/ -R
```

2. Change the **Repo** setting on the **TemplateSync** tab to match the export directory `/usr/share/templates_dir/`.
3. Export templates from your Satellite Server to a local directory:

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://satellite.example.com/api/v2/templates/export \
```

```
-X POST \
{"message":"Success"}
```

4. Import templates to Satellite Server after their content was changed:

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://satellite.example.com/api/v2/templates/import \
-X POST
{"message":"Success"}
```

Note that templates provided by Satellite are locked and you cannot import them by default. To overwrite this behavior, change the **Force import** setting in the **TemplateSync** menu to **yes** or add the **force** parameter **-d '{ "force": "true" }'** to the import command.



#### NOTE

You can override default API settings by specifying them in the request with the **-d** parameter. The following example exports templates to the **git.example.com/templates** repository:

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://satellite.example.com/api/v2/templates/export \
-X POST \
-d '{"repo":"git.example.com/templates"}"
```

## D.4. ADVANCED GIT CONFIGURATION

You can perform additional Git configuration for the TemplateSync plug-in using the command line or editing the **.gitconfig** file.

### Accepting a self-signed Git certificate

If you are using a self-signed certificate authentication on your Git server, validate the certificate with the **git config http.sslCAPath** command.

For example, the following command verifies a self-signed certificate stored in **/cert/cert.pem**:

```
# sudo -u foreman git config --global http.sslCAPath cert/cert.pem
```

For a complete list of advanced options, see the **git-config** manual page.

## D.5. UNINSTALLING THE PLUG-IN

To avoid errors after uninstallation:

1. Disable the plug-in using the Satellite installer:

```
# satellite-installer --no-enable-foreman-plugin-templates
```

2. Clean custom data of the plug-in. The command does not affect any templates that you created.

```
# foreman-rake templates:cleanup
```

3. Uninstall the plug-in:

```
# yum remove tfm-rubygem-foreman_templates
```