Red Hat Satellite 6.5

Content Management Guide

A guide to managing content from Red Hat and custom sources

Last Updated: 2020-04-14

Red Hat Satellite 6.5 Content Management Guide

A guide to managing content from Red Hat and custom sources

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

Legal Notice

Abstract

Use this guide to understand and manage content in Satellite 6. Examples of such content include RPM files, ISO images, and Puppet modules. 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

CHAPTER 1. INTRODUCTION 5
1.1. CONTENT MANAGEMENT TYPES OVERVIEW 5

CHAPTER 2. MANAGING ORGANIZATIONS 6
2.1. CREATING AN ORGANIZATION 6
2.2. SETTING THE ORGANIZATION CONTEXT 7
2.3. CREATING AN ORGANIZATION DEBUG CERTIFICATE 7
2.4. BROWSING REPOSITORY CONTENT USING AN ORGANIZATION DEBUG CERTIFICATE 8
2.5. DELETING AN ORGANIZATION 9

CHAPTER 3. MANAGING LOCATIONS 10
3.1. CREATING A LOCATION 10
3.2. SETTING THE LOCATION CONTEXT 10
3.3. DELETING A LOCATION 11

CHAPTER 4. MANAGING SUBSCRIPTIONS 12
4.1. IMPORTING A SUBSCRIPTION MANIFEST INTO SATELLITE SERVER 12
4.2. LOCATING A SUBSCRIPTION IN THE SATELLITE WEB UI 13
4.3. ADDING SUBSCRIPTIONS TO SUBSCRIPTION ALLOCATIONS IN THE SATELLITE WEB UI 13
4.4. REMOVING SUBSCRIPTIONS FROM SUBSCRIPTION ALLOCATIONS IN THE SATELLITE WEB UI 14
4.5. UPDATING AND REFRESHING SUBSCRIPTION MANIFESTS 14
4.6. ATTACHING SUBSCRIPTIONS TO CONTENT HOSTS 15
4.7. BULK UPDATING CONTENT HOSTS' SUBSCRIPTIONS 16

CHAPTER 5. IMPORTING RED HAT CONTENT 18
5.1. PRODUCTS IN RED HAT SATELLITE 18
5.2. CONTENT SYNCHRONIZATION OVERVIEW 18
5.3. DISABLING THE GLOBAL HTTP PROXY 18
5.4. DOWNLOAD POLICIES OVERVIEW 18
5.5. CHANGING THE DEFAULT DOWNLOAD POLICY 19
5.6. CHANGING THE DOWNLOAD POLICY FOR A REPOSITORY 20
5.7. ENABLING RED HAT REPOSITORIES 20
5.8. SYNCHRONIZING RED HAT REPOSITORIES 21
5.9. SYNCHRONIZING ALL REPOSITORIES IN AN ORGANIZATION 22
5.10. RECOVERING A REPOSITORY 23
5.11. LIMITING SYNCHRONIZATION SPEED 24
5.12. CREATING A SYNCHRONIZATION PLAN 24

CHAPTER 6. IMPORTING CUSTOM CONTENT 26
6.1. USING CUSTOM PRODUCTS IN SATELLITE 26
6.2. IMPORTING CUSTOM SSL CERTIFICATES 26
6.3. IMPORTING A CUSTOM GPG KEY 26
6.4. CREATING A CUSTOM PRODUCT 28
6.5. ADDING A CUSTOM RPM REPOSITORY 28

CHAPTER 7. MANAGING APPLICATION LIFE CYCLES 30
7.1. APPLICATION LIFE CYCLE OVERVIEW 30
7.2. PROMOTING CONTENT ACROSS THE APPLICATION LIFE CYCLE 31
7.3. CREATING A LIFE CYCLE ENVIRONMENT PATH 32
7.4. REMOVING LIFE CYCLE ENVIRONMENTS FROM SATELLITE SERVER 33
7.5. REMOVING LIFE CYCLE ENVIRONMENTS FROM CAPSULE SERVER 34
7.6. ADDING LIFE CYCLE ENVIRONMENTS TO CAPSULE SERVERS 35

CHAPTER 8. MANAGING CONTENT VIEWS 37
8.1. CREATING A CONTENT VIEW 38
8.2. VIEWING MODULE STREAMS 39
8.3. CREATING A CONTENT VIEW WITH A PUPPET MODULE 39
8.4. PROMOTING A CONTENT VIEW 39
8.5. COMPOSITE CONTENT VIEWS OVERVIEW 41
8.6. CREATING A COMPOSITE CONTENT VIEW 42
8.7. CONTENT FILTER OVERVIEW 43
8.8. CONTENT FILTER EXAMPLES 45
8.9. CREATING A CONTENT FILTER 46

CHAPTER 9. SYNCHRONIZING CONTENT BETWEEN SATELLITE SERVERS 48
9.1. EXPORTING A CONTENT VIEW VERSION 48
9.2. IMPORTING A CONTENT VIEW VERSION 49

CHAPTER 10. MANAGING ACTIVATION KEYS 51
10.1. CREATING AN ACTIVATION KEY 51
10.2. UPDATING SUBSCRIPTIONS ASSOCIATED WITH AN ACTIVATION KEY 53
10.3. UPDATING SUBSCRIPTIONS ASSOCIATED WITH AN ACTIVATION KEY USING A CSV FILE 54
10.4. USING ACTIVATION KEYS FOR HOST REGISTRATION 56
10.5. ENABLING AUTO-ATTACH 57
10.6. SETTING THE SERVICE LEVEL 58

CHAPTER 11. MANAGING ERRATA 59
11.1. INSPECTING AVAILABLE ERRATA 59
11.2. SUBSCRIBING TO ERRATA NOTIFICATIONS 61
11.3. LIMITATIONS TO REPOSITORY DEPENDENCY RESOLUTION 61
11.4. CREATING A CONTENT VIEW FILTER FOR ERRATA 62
11.5. ADDING ERRATA TO AN INCREMENTAL CONTENT VIEW 63
11.6. APPLYING ERRATA TO A HOST 64
11.7. APPLYING ERRATA TO MULTIPLE HOSTS 65

CHAPTER 12. MANAGING OSTREE CONTENT 68
12.1. SELECTING RED HAT OSTREE CONTENT TO SYNCHRONIZE 68
12.2. IMPORTING CUSTOM OSTREE CONTENT 69
12.3. MANAGING OSTREE CONTENT WITH CONTENT VIEWS 70

CHAPTER 13. MANAGING ISO IMAGES 72
13.1. IMPORTING ISO IMAGES FROM RED HAT 72
13.2. IMPORTING INDIVIDUAL ISO IMAGES AND FILES 73

CHAPTER 14. MANAGING CUSTOM FILE TYPE CONTENT 75
14.1. CREATING A CUSTOM FILE TYPE REPOSITORY IN RED HAT SATELLITE 75
14.2. CREATING A CUSTOM FILE TYPE REPOSITORY IN A LOCAL DIRECTORY 77
14.3. CREATING A REMOTE FILE TYPE REPOSITORY 79
14.4. UPLOADING FILES TO A CUSTOM FILE TYPE REPOSITORY IN RED HAT SATELLITE 80
14.5. DOWNLOADING FILES TO A HOST FROM A CUSTOM FILE TYPE REPOSITORY IN RED HAT SATELLITE 81

CHAPTER 15. MANAGING CUSTOM PUPPET CONTENT 83
15.1. CREATING A CUSTOM PUPPET REPOSITORY 83
15.2. MANAGING INDIVIDUAL PUPPET MODULES 83
15.3. SYNCHRONIZING PUPPET REPOSITORIES 84
15.4. SYNCHRONIZING PUPPET MODULES FROM A GIT REPOSITORY 85

APPENDIX A. USING AN NFS SHARE FOR CONTENT STORAGE 87

APPENDIX B. IMPORTING CONTENT ISO IMAGES INTO A DISCONNECTED SATELLITE 89
B.1. IMPORTING CONTENT ISO IMAGES 89
B.2. IMPORTING KICKSTART REPOSITORIES 90

APPENDIX C. IMPORTING CONTENT ISOS INTO A CONNECTED SATELLITE 95

APPENDIX D. SYNCHRONIZING TEMPLATES WITH GIT 98
D.1. ENABLING THE TEMPLATESYNC PLUG-IN 98
D.2. CONFIGURING THE TEMPLATESYNC PLUG-IN 98
D.3. IMPORTING AND EXPORTING TEMPLATES 99
D.3.1. Synchronizing Templates with a Git repository 100
D.3.2. Synchronizing templates with a local directory 101
D.4. ADVANCED GIT CONFIGURATION 102
D.5. UNINSTALLING THE PLUG-IN 102

CHAPTER 1. INTRODUCTION

In the context of Satellite 6, *content* is defined as the software installed on systems. This includes, but is not limited to, the base operating system, middleware services, and end-user applications. With Red Hat Satellite 6, you can manage the various types of content for Red Hat Enterprise Linux systems at every stage of the software life cycle.

Red Hat Satellite 6 manages the following content:

Subscription management

This provides organizations with a method to manage their Red Hat subscription information.

Content management

This provides organizations with a method to store Red Hat content and organize it in various ways.

1.1. CONTENT MANAGEMENT TYPES OVERVIEW

With Red Hat Satellite 6, you can manage the following Red Hat content types:

RPM Packages

Import RPM files from repositories related to your Red Hat subscriptions. 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

Import the kickstart trees for creating a 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 as well as the ability to create your own, which are used to provision systems and customize the installation.

You can also manage custom content in Red Hat Satellite 6. For example:

ISO and KVM Images

Download and manage media for installation and provisioning. For example, Satellite downloads, stores and manages ISO images and guest images for specific Red Hat Enterprise Linux and non-Red Hat operating systems.

Puppet Modules

You can upload Puppet modules alongside RPM content so that Puppet can configure the system's state after provisioning. Users can also manage Puppet classes and parameters as part of the provisioning process.

OSTree

You can import OSTree branches and publish this content to an HTTP location.

You can use the procedure to add custom content for any type of content you require, for example, SSL certificates and OVAL files.

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, you can 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, you can create organizations for the company and each of the business units it owns. This keeps the system infrastructure for each separate. You can then assign content to each organization based on its 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, you can create an organization for the company's own system infrastructure and then an organization for each external business. You can then assign content to each organization where necessary.

A default installation of Red Hat Satellite 6 has a default organization called **Default_Organization**.

New Users

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. The next time the user logs on to Satellite, the user's account has the correct system rights.

2.1. CREATING AN ORGANIZATION

Use this procedure to create an organization.

Procedure

To create an organization, complete the following steps:

1. In the Satellite web UI, navigate to **Administer** > **Organizations**.
2. Click **New Organization**.
3. In the **Name** field, enter a name for the organization.
4. In the **Label** field, enter 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.
5. Optional: in the **Description** field, enter a description for the organization.
6. Click **Submit**.
7. If you have hosts with no organization assigned, select the hosts that you want to add to the organization, then click **Proceed to Edit**

8. In the **Edit** page, assign the infrastructure resources that you want to add 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.

9. 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.

Procedure

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

If you require a debug certificate for your organization, use the following procedure.

Procedure

To create a debug certificate for an organization, complete the following steps:

1. In the Satellite web UI, navigate to **Administer** > **Organizations**.

2. Select an organization that you want to generate a debug certificate for.

3. Click **Generate and Download**.

4. Save the certificate file in a secure location.

Debug Certificates for Provisioning Templates

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. BROWSING REPOSITORY CONTENT USING AN ORGANIZATION DEBUG CERTIFICATE

You can view an organization’s repository content using a web browser or using the API if you have a debug certificate for that organization.

Prerequisites

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-----**, into a **key.pem** file.
4. Copy the contents of the file from **-----BEGIN CERTIFICATE-----** to **-----END CERTIFICATE-----**, into a **cert.pem** file.

Procedure

To use a browser, you must first convert the X.509 certificate to a format your browser supports and then import the certificate.

For Firefox Users

To use an organization debug certificate in Firefox, complete the following steps:

1. To create a PKCS12 format certificate, enter the following command:

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

2. In the Firefox browser, navigate to **Edit** > **Preferences** > **Advanced Tab**.
3. Select **View Certificates**, and click the **Your Certificates** tab.
4. Click **Import** and select the **.pfx** file to load.
5. In the address bar, enter a URL in the following format to browse for repositories:

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

   Pulp uses the organization label, therefore, you must enter the organization label into the URL.

For CURL Users

To use the organization debug certificate with CURL, 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/6.5/os/
```

Ensure that 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. Do not 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 at any given time.

Procedure

To delete an organization, complete the following steps:

1. In the Satellite web UI, navigate to **Administer** > **Organizations**.
2. From the list to the right of the name of the organization you want to delete, select **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. Organizations and locations have the following conceptual differences:

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

3.1. CREATING A LOCATION

Use this procedure to create a location so that you can manage your hosts and resources by location.

Procedure

To create a location, complete the following steps:

1. In the Satellite web UI, navigate to **Administer > Locations**.
2. Click **New Location**.
3. Optional: from the **Parent** list, select a parent location. This creates a location hierarchy.
4. In the **Name** field, enter a name for the location.
5. Optional: in the **Description** field, enter a description for the location.
6. Click **Submit**.
7. If you have hosts with no location assigned, add any hosts that you want to assign to the new location, then click **Proceed to Edit**
8. Assign any infrastructure resources that you want to add 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.
9. Click **Submit** to save your changes.

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.

Procedure

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 displays **Any Location**. Click **Any location** 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. Do not 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 at any given time.

Procedure

To delete a location, complete the following steps:

1. In the Satellite web UI, 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). Satellite 6 requires a Subscription Manifest to find, access, and download content from the corresponding repositories. You must have a Subscription Manifest containing a subscription allocation for each organization on Satellite Server. All subscription information is available in your Red Hat Customer Portal account.

Before you can complete the tasks in this chapter, you must create a Subscription Manifest in the Customer Portal.

To create, manage, and export a Subscription Manifest in the Customer Portal, see Using Manifests in the *Using Red Hat Subscription Management* guide.

Use this chapter to import a Subscription Manifest and manage the manifest within the Satellite web UI.

Subscription Allocations and 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 Satellite Server. The advantage of this is that each organization maintains separate subscriptions so that you can support multiple organizations, each with their own Red Hat accounts.

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.

Manually attach the future-dated subscriptions to your content hosts before the current subscriptions expire. Do not rely on the auto-attach method because this method is designed for a different purpose and might not work. For more information, see Section 4.6, “Attaching Subscriptions to Content Hosts” .

4.1. IMPORTING A SUBSCRIPTION MANIFEST INTO SATELLITE SERVER

Use the following procedure to import a Subscription Manifest into Satellite Server.

Prerequisite

You must have a Subscription Manifest file exported from the Customer Portal. For more information, see Using Manifests in the *Using Red Hat Subscription Management* guide.

Procedure

To import a Subscription Manifest, complete the following steps:

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

5. Navigate to the location that contains the Subscription Manifest file, and then click **Open**. If the Manage Manifest window does not close automatically, click **Close** to return to the Subscriptions window.

For CLI Users

1. Copy the Subscription Manifest file to Satellite Server:

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

2. Connect to 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"
```

You can now enable repositories and import Red Hat content. For more information, see Chapter 5, *Importing Red Hat Content*.

4.2. LOCATING A SUBSCRIPTION IN THE SATELLITE WEB UI

When you import a Subscription Manifest into Satellite Server, the subscriptions from your manifest are listed in the Subscriptions window. 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 imported to Satellite Server. For more information, see Section 4.1, "Importing a Subscription Manifest into Satellite Server" .

Procedure

To locate a subscription, complete the following steps:

1. In the Satellite web UI, ensure the context is set to the organization you want to use.
2. Navigate to **Content** > **Subscriptions**.
3. In the Subscriptions window, click the **Search** field to view the list of search criteria for building your search query.
4. Select 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 **=** character. If you select **>** and press the space bar, another list of automatic options appears. You can also enter your own criteria.

4.3. ADDING SUBSCRIPTIONS TO SUBSCRIPTION ALLOCATIONS IN THE SATELLITE WEB UI

Use the following procedure to add subscriptions to a subscription allocation in the Satellite web UI.

Prerequisite

You must have a Subscription Manifest file imported to Satellite Server. For more information, see Section 4.1, "Importing a Subscription Manifest into Satellite Server" .

Procedure

To add subscriptions to a subscription allocation, complete the following steps:

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

4.4. REMOVING SUBSCRIPTIONS FROM SUBSCRIPTION ALLOCATIONS IN THE SATELLITE WEB UI

Use the following procedure 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 Satellite Server. For more information, see Section 4.1, "Importing a Subscription Manifest into Satellite Server" .

Procedure

To remove subscriptions, complete the following steps:

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.

4.5. 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 refresh the manifest directly in the Satellite web UI. Alternatively, you can import an updated manifest that contains the changes.

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. In the Subscriptions window, click **Manage Manifest**.
4. In the Manage Manifest window, click **Refresh**.

4.6. ATTACHING SUBSCRIPTIONS TO CONTENT HOSTS

Using activation keys is the main method to attach subscriptions to content hosts during the provisioning process. 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, use the following procedure.

For more information about updating multiple hosts, see Section 4.7, “Bulk Updating Content Hosts' Subscriptions”.

For more information about activation keys, see Chapter 10, *Managing Activation Keys*.

Smart Management Subscriptions

In Satellite 6, you must maintain a Red Hat Enterprise Linux Smart Management subscription for every Red Hat Enterprise Linux host that you want to manage.

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.

Prerequisite

You must have a Subscription Manifest file imported to Satellite Server.

Procedure

To attach subscriptions to content hosts, complete the following steps:

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 that you want to add or remove and click **Add Selected** or **Remove Selected** as required.

7. Click **Done** to save the changes.

For CLI Users

1. Connect to 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
```

4.7. BULK UPDATING CONTENT HOSTS' SUBSCRIPTIONS

Use this procedure for post-installation changes to multiple content hosts at the same time. You can use the Satellite web UI and the filter function to select the content hosts that you want to change, or use the Hammer CLI tool's CSV file export function, edit the configuration settings in the CSV file, and upload the changes.

Procedure

To update multiple content hosts, complete the following steps:

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. Make a backup of the file:

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

_

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 CSV file. For example:

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

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

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

5. 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

This section describes how to use products and repositories in Satellite, and to create synchronization plans to ensure your Satellite content remains up to date with the content on the Red Hat Content delivery network (CDN).

5.1. PRODUCTS IN RED HAT SATELLITE

In Satellite, a *Product* is an organizational unit to group multiple repositories together. For example, Red Hat Enterprise Linux Server is a *Product* in Satellite, the repositories for that product consist of different versions, different architectures, and different add-ons. Using Products ensures repositories that depend on each other are synchronized together. For Red Hat repositories, products are created automatically after enabling the repository.

5.2. CONTENT SYNCHRONIZATION OVERVIEW

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. 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 are up to date with the CDN's repositories.

You can perform an initial synchronization using ISO images. For more information about using content ISOs, see Appendix C, *Importing Content ISOs into a Connected Satellite* .

5.3. 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. DOWNLOAD POLICIES OVERVIEW

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

Satellite Server has the following policies:

Immediate

Satellite Server downloads all metadata and packages during synchronization.

On Demand

Satellite Server downloads only 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 **On Demand** and **Background** policies act as a *Lazy Synchronization* feature because they save time synchronizing content. The lazy synchronization feature must be used only for **yum** repositories. You can add the packages to Content Views and promote 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. Capsule Server fetches and stores packages only on the file system when directly connected clients request them. When you use an **On Demand** download policy, content is downloaded from Satellite Server if it is not available on 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.

5.5. CHANGING THE DEFAULT DOWNLOAD POLICY

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

Procedure

To change the default download policy for repositories, complete the following steps:

1. In the Satellite web UI, navigate to **Administer** > **Settings**.
2. Click the **Content** tab and locate the **Default Repository 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**, **background**, enter the following command:

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

5.6. CHANGING THE DOWNLOAD POLICY FOR A REPOSITORY

You can also set the download policy for a repository.

Procedure

To set the download policy for a repository, complete the following steps:

1. In the web UI, navigate to **Content** > **Products**, and click the required product name.
2. On the **Repositories** tab, 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**, **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
   ```

5.7. ENABLING RED HAT REPOSITORIES

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. For Red Hat Enterprise Linux 8, you must enable both AppStream and BaseOS repositories.

Disconnected Satellite

If you use Disconnected Satellite Server, you must 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 ISO Images into a Disconnected Satellite* .

Repository Versioning

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 Red Hat Enterprise Linux 8 Clients

To provision Red Hat Enterprise Linux 8 clients, you require the **Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMS)** and **Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)**repositories.

For Red Hat Enterprise Linux 7 Clients

To provision Red Hat Enterprise Linux 8 clients, you require the **Red Hat Enterprise Linux 7 Server (RPMs)** repository.

Procedure

1. In the Satellite web UI, navigate to **Content** > **Red Hat Repositories**.
2. To find repositories, either enter the repository name, or toggle the **Recommended Repositories** button to the on position to view a list of repositories that you require.
3. In the Available Repositories pane, click a repository to expand the repository set.
4. Click the **Enable** icon next to the base architecture and release version that you want.

For CLI Users

1. To search for your product, enter the following command:

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

2. List the repository set for the product:

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

3. Enable the repository using either the name or ID number. Include the release version, for example,**7Server** and base architecture, for example, **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 "My_Organization"
```

5.8. SYNCHRONIZING RED HAT REPOSITORIES

Synchronize the repositories with the Red Hat CDN's repositories.

For Web UI Users

1. In the Satellite web UI, navigate to **Content** > **Products** and select the product that contains the repositories that you want to synchronize.
2. Select the repositories that you want to synchronize and click **Sync Now**.

To view the progress of the synchronization in the web UI, navigate to **Content** > **Sync Status** and expand the corresponding product or repository tree.

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 "My_Organization"
```

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

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

Create a synchronization plan to ensure updates on a regular basis.

5.9. SYNCHRONIZING ALL REPOSITORIES IN AN ORGANIZATION

Use this procedure to synchronize all repositories within an organization.

Procedure

To synchronize all repositories within an organization, run the following Bash script on your Satellite Server:

```
ORG="Your_Organization"

for i in $(hammer --no-headers --csv repository list --organization $ORG | awk -F, {'print $1'})
do
  hammer repository synchronize --id ${i} --organization $ORG --async
done
```

5.10. RECOVERING A REPOSITORY

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

Optimized Sync

Synchronizes the repository bypassing RPMs that have no detected differences from the upstream RPMs.

Complete Sync

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

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 only for **yum** repositories. 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.

Procedure

To synchronize a specific repository with an advanced option, complete the following steps:

1. In the Satellite web UI, navigate to **Content** > **Products**.
2. Select the product containing the corrupted repository.
3. Select the name of a repository you want to synchronize.
4. From the **Select Action** menu, select **Advanced Sync**.
5. Select the option and click **Sync**.

For CLI users

1. Obtain a list of repository IDs:

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

5.11. LIMITING SYNCHRONIZATION SPEED

You can control the speed of synchronization to avoid exhaustion of available bandwidth and to prevent other performance issues. This is done by configuring **PULP_CONCURRENCY** and **max_speed** parameters. Note that these settings are overwritten on an upgrade. Back up any 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. 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
   }
   ```

3. Verify the syntax of the file after editing:

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

4. Restart the **satellite-maintain** services to apply the changes:

   ```
   # satellite-maintain service restart
   ```

5.12. CREATING A SYNCHRONIZATION PLAN

A synchronization plan checks and updates the content at a scheduled date and time. In Red Hat Satellite 6, you can create a synchronization plan and assign products to the plan.

Procedure

To create a synchronization plan, complete the following steps:

1. In the Satellite web UI, navigate to **Content** > **Sync Plans** and click **New Sync Plan**.
2. In the **Name** field, enter a name for the plan.
3. In the **Description** field, enter a description of the plan.
4. From the **Interval** list, select the interval at which you want the plan to run.
5. From the **Start Date** and **Start Time** lists, select when to start running the synchronization plan.
6. Click **Save**.
7. Click the **Products** tab, then click **Add**. Select the **Red Hat Enterprise Linux Server** product and click **Add Selected**.

For CLI Users

1. To create the synchronization plan, enter the following command:

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

2. 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 "My_Organization"
```

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 12, *Managing OSTree Content*
- Chapter 13, *Managing ISO Images*
- Chapter 14, *Managing Custom File Type Content*
- Chapter 15, *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+COH9kMqXHi3M5UNzbp5WhCdUpISXjjpU
   XIFFWBPuBfyr/FKRknFH15P+9kLZLxCpVZZLsweLWCuw+JKCMmnA
   =F6VG
   -----END PGP PUBLIC KEY BLOCK-----

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

   mQINBFw467UBEACmREzDeK/kuScCmfJfHJa0Wgh/2fbJLLt3KSvsgDhORIptf+PP
   OTFDlKuLkJx99ZYG5xMnBG47C7ByoMec1j94YeXczuBbynOyyPlvduma/zf8oB9e
   Wl5GnzcLGAnUSRamfqGUWcyMMinHHIKlc1X1P4I=
   =WPpI
   -----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 **Create 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

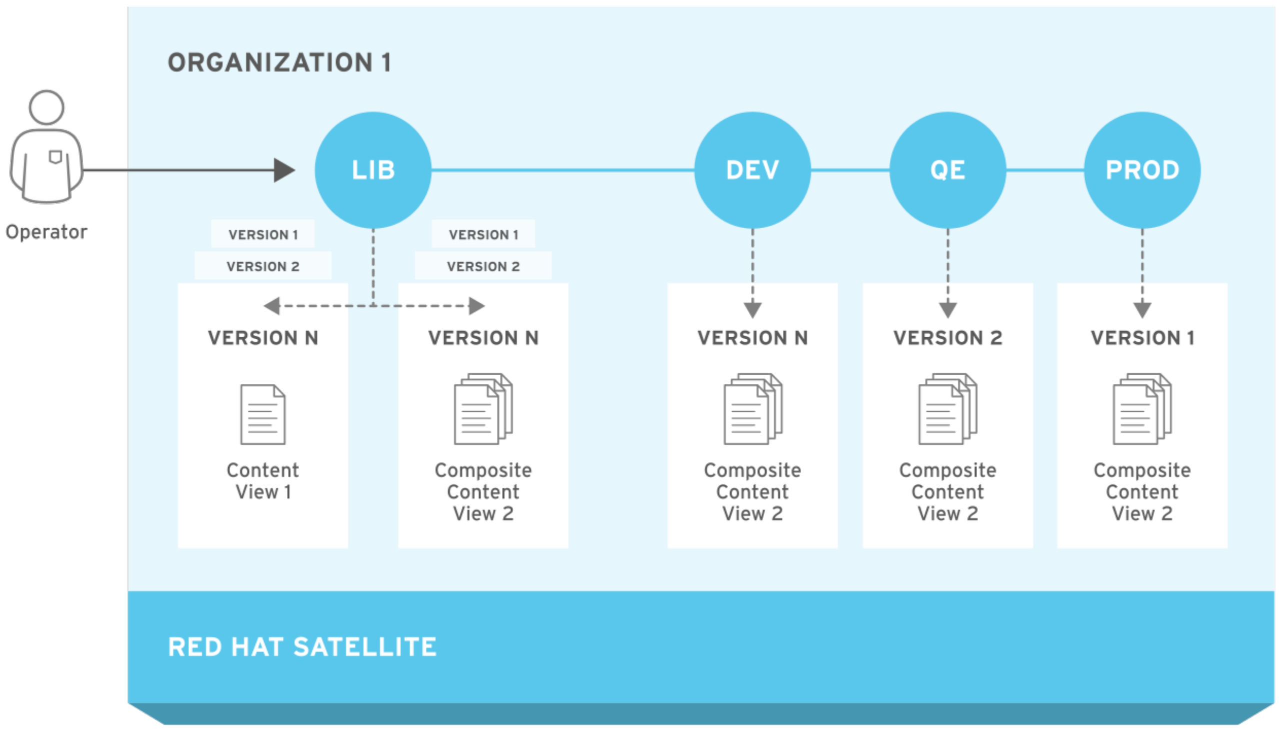
Red Hat Satellite 6 provides methods to customize each application life cycle stage so that it suits your specifications.

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
example_software-1.1-0.noarch.rpm	*example_software*-1.0-0.noarch.rpm	*example_software*-1.0-0.noarch.rpm

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
example_software-1.1-0.noarch.rpm	***example_software*-1.1-0.noarch.rpm**	*example_software*-1.0-0.noarch.rpm

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
exampleware-2.0-0.noarch.rpm	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	***example_software*-1.1-0.noarch.rpm**

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	***example_software*-2.0-0.noarch.rpm**	*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	**exampleware-2.0-0.noarch.rpm**

For more information, see Section 8.4, “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**.

6. 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

1. 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"
   ```

2. 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"
   ```

3. 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.

Procedure

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

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

For CLI Users

1. 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"
   ```

2. 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 repositories. 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 - *example_software*-1.1-0.noarch.rpm	Version 1 - *example_software*-1.0-0.noarch.rpm	Version 1 - *example_software*-1.0-0.noarch.rpm

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 - *example_software*-1.1-0.noarch.rpm	**Version 2 - *example_software*-1.1-0.noarch.rpm**	Version 1 - *example_software*-1.0-0.noarch.rpm

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 8.6, “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. Click **Save**.
8. Optional: to force metadata regeneration on Yum repositories, from the **Actions** list for your Content View versions, select **Regenerate Repository Metadata**.

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 Hosts in the *Managing Hosts* guide.

Creating a Content View with Hammer CLI

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. VIEWING MODULE STREAMS

In Satellite, you can view the module streams of the repositories in your Content Views.

Procedure

To view module streams for the repositories in your content view, complete the following steps:

1. In the Satellite web UI, navigate to **Content** > **Content Views**, and select the Content View that contains the modules you want to view.
2. Click the **Versions** tab and select the Content View version that you want to view.
3. Click the **Module Streams** tab to view the module streams that are available for the Content View.
4. Use the **Filter** field to refine the list of modules.
5. To view the information about the module, click the module.

8.3. 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 Hosts in the *Managing Hosts* guide.

8.4. 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 Hosts in the *Managing Hosts* guide.

8.5. 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:

example_software
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 - *example_software* on PostgreSQL
example_software (Application)
PostgreSQL (Database)
Red Hat Enterprise Linux (Operating System)

Example Composite Content View for a MariaDB:

Composite Content View 2 - *example_software* on MariaDB
example_software (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.

8.6. 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 your existing Content Views:

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

2. Create a new Composite Content View. 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" \
--organization "My_Organization"
```

3. Add a component Content View to the Composite Content View. You must include the Content View Version ID and use the **--latest** option. To include multiple component Content Views to the Composite Content View, repeat this step for every Content View you want to include:

```
# hammer content-view component add \
--component-content-view-id Content_View_Version_ID \
--latest \
--composite-content-view "Example_Composite_Content_View"
```

4. Publish the Composite Content View:

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

5. 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"
```

8.7. 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
Include	You start with no content, then select which content to add from the selected repositories. Use this filter to combine multiple content items.
Exclude	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.

8.8. 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"

8.9. CREATING A CONTENT FILTER

Use this procedure to create a content filter. For examples of how to build a filter, see Section 8.8, "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.
9. 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

1. 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"
```

2. 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"
```

3. Publish the Content View:

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

4. 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. SYNCHRONIZING CONTENT BETWEEN SATELLITE SERVERS

Red Hat Satellite 6.5 uses Inter-Satellite Synchronization (ISS) to synchronize content Satellite Servers, or between organizations on Satellite Server.

You can use ISS in the following scenarios:

- If you have both connected and disconnected Satellite Servers, and want to copy content from the connected servers to the disconnected servers. For example, you require complete isolation of management infrastructure for security or other purposes.
- If you want to copy some but not all content from your Satellite Server to other Satellite Servers. For example, you have Content Views that your IT department validates on Satellite Server, and you want to copy content from those Content Views to other Satellite Servers.
- If you want to clone a Content View from one organization to another organization on Satellite Server.

You cannot use ISS to synchronize content from Satellite Server to Capsule Server. Capsule Server supports synchronization natively. For more information, see Capsule Server Overview in *Planning for Red Hat Satellite 6*.

9.1. EXPORTING A CONTENT VIEW VERSION

You can export a version of a Content View to an archive file from Satellite Server and use this archive file to create the same Content View version on another Satellite Server or on another Satellite Server organization. Satellite does not export composite Content Views. The exported archive file contains the following data:

- A JSON file containing Content View version metadata
- An archive file containing all the repositories included into the Content View version

Satellite Server exports only RPM and kickstart files added to a version of a Content View. Satellite does not export the following content:

- Puppet content
- Docker content
- OSTree content
- Content View definitions and metadata, such as package filters.

Changes to the hammer content-view version export command

The new **hammer content-view version export** and **hammer content-view version import** commands work differently from the commands in the previous versions of Satellite. The old feature is still available with the **hammer content-view version export-legacy** command. The old feature has the following functionality that does not exist in the new feature:

1. You can patch a disconnected Satellite Server from a connected Satellite Server directly. **hammer content-view version export-legacy** exports the CDN structure, therefore, you do not have to use a DVD ISO from the Red Hat Customer Portal.

2. When exporting a Content View that contains non-yum content, **hammer content-view version export-legacy** skips the non-yum content and exports the Content View, while **hammer content-view version export** prompts you to remove a non-yum repository and fails.

For more information about using the old feature, see Synchronizing Content Between Satellite Servers in the Satellite 6.4 Content Management Guide.

Prerequisites

To export a Content View, ensure that the Satellite Server where you want to export meets the following conditions:

- Ensure that the export directory has free storage space to accommodate the 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, "Download Policies Overview".
- Ensure that you clear the **Mirror on Sync** check box for the repositories that you import on the repository settings page.
- Ensure that you synchronize Products that you export to the required date.

To Export a Content View Version:

1. List Content Views to determine the ID of a Content View version you want to export:

```
# hammer content-view version list \
--organization "Default Organization"
```

2. Export the version of a Content View. Specify the directory where to store the export with the **--export-dir** option and the ID of the Content View version that you export with the **--id** option. The **pulp_export_destination** setting does not work for this procedure.

```
# hammer content-view version export --export-dir export_directory \
--id content_view_version_ID
```

3. Verify that the archive containing the exported version of a Content View is located in the export directory:

```
# ls export_directory
export-1.tar
```

9.2. IMPORTING A CONTENT VIEW VERSION

You can use the archive that the **hammer content-view version export** command outputs to create a version of a Content View with the same content as the exported Content View version. For more information about exporting a Content View version, see Section 9.1, "Exporting a Content View Version".

When you import a Content View version, it has the same major and minor version numbers and contains the same repositories with the same packages and errata. You can customize the version numbers by changing the **major** and **minor** settings in the **json** file that is located in the exported archive.

Prerequisites

To import a Content View, ensure that the Satellite Server where you want to import meets the following conditions:

- If you want to import a Content View to a disconnected Satellite, you must download the content ISOs from the Red Hat Customer Portal and import them into your Satellite Server. For more information, see Appendix B, *Importing Content ISO Images into a Disconnected Satellite* .
- Ensure that you set download policy to **Immediate** for all repositories within the Content View you export. For more information, see Section 5.4, "Download Policies Overview".
- Ensure that you clear the **Mirror on Sync** check box for the repositories that you import on the repository settings page.

Procedure

1. Copy the archived file with the exported Content View version to the **/var/lib/pulp/katello-export** directory on the Satellite Server where you want to import.
2. On the Satellite Server where you want to import, create a Content View with the same name and label as the exported Content View. For more information, see Creating a Content View with Hammer CLI.
3. Ensure that you enable the repositories that the Products in the exported Content View version includes. For more information, see Section 5.7, "Enabling Red Hat Repositories".
4. In the Satellite web UI, navigate to **Content** > **Products**, click the **Yum content** tab and add the same **Yum** content that the exported Content View version includes.
5. Until BZ#1745081 is resolved, navigate to the **/var/lib/pulp/katello-export** directory:

```
# cd /var/lib/pulp/katello-export
```

6. To import the Content View version to Satellite Server, enter the following command:

```
# hammer content-view version import \
--export-tar /var/lib/pulp/katello-export/exported_CV_archive.tar \
--organization-id Your_Organization_ID
```

Note that until BZ#1745081 is resolved, you must enter the full path **/var/lib/pulp/katello-export/**. Relative paths do not work.

7. To verify that you import the Content View version successfully, list Content Views for your organization:

```
# hammer content-view version list --organization "Your_Organization"
```

CHAPTER 10. 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.

10.1. CREATING AN ACTIVATION KEY

You can use activation keys to define a specific set of subscriptions to attach to hosts during registration. The subscriptions that you add to an activation key must be available within the associated Content View.

Subscription Manager attaches subscriptions differently depending on the following 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 the Content View associated with the activation key. 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. If you want to use this activation key to register hosts, the Content View must contain the Satellite Tools repository because it is required to install the **katello-agent**.
7. Click **Save** and when your new activation key appears in the Activation Keys window, click the name to edit.

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 6.5 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-6.5-rpms \
--value 1 \
--organization "My_Organization"
```

10.2. 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 4.7, "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.
7. Click the **Details** tab, select a Content View for this activation key, and then click **Save**.

For CLI Users

1. List the subscriptions that the activation key currently contains:

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

2. 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.

3. 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.

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 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.

10.3. 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
```

10.4. 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 6.5 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**.

10.5. 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
```

_

10.6. 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 11. 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
: An erratum that applies to one or more content hosts, which means it updates packages present on the content host. Although these errata apply to content hosts, until their state changes to **Installable**, the errata are not ready to be installed. Installable errata are automatically applicable.

Installable
: An erratum that applies to one or more content hosts and is available to install on the content host. Installable errata are available to a content host from life cycle environment and the associated Content View, but are not yet installed.

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

11.1. 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 11.1, "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.

3. 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 11.7, "Applying Errata to Multiple Hosts" .
   - 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 11.1. Parameters Available for Errata Search

Parameter	Description	Example
bug	Search by the Bugzilla number.	*bug = 1172165*
cve	Search by the CVE number.	*cve = CVE-2015-0235*
id	Search by the errata ID. The auto-suggest system displays a list of available IDs as you type.	*id = RHBA-2014:2004*
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.	*issued < "Jan 12,2015"*
package	Search by the full package build name. The auto-suggest system displays a list of available packages as you type.	*package = glib2-2.22.5-6.el6.i686*

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

11.2. 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.

11.3. LIMITATIONS TO REPOSITORY DEPENDENCY RESOLUTION

There are a number of challenges to solving repository dependencies in Satellite 6. This is a known issue. For more information, see BZ#1508169, BZ#1640420, BZ#1508169, and BZ#1629462. With Satellite, using incremental updates to your Content Views solves some repository dependency problems. However, dependency resolution at a repository level still remains problematic on occasion.

When a repository update becomes available with a new dependency, Satellite retrieves the newest version of the package to solve the dependency, even if there are older versions available in the existing repository package. This can create further dependency resolution problems when installing packages.

Example scenario

A repository on your client has the package **example_repository-1.0** with the dependency **example_repository-libs-1.0**. The repository also has another package **example_tools-1.0**.

A security erratum becomes available with the package **example_tools-1.1**. The **example_tools-1.1** package requires the **example_repository-libs-1.1** package as a dependency.

After an incremental Content View update, the **example_tools-1.1**, **example_tools-1.0**, and **example_repository-libs-1.1** are now in the repository. The repository also has the packages **example_repository-1.0** and **example_repository-libs-1.0**. Note that the incremental update to the Content View did not add the package **example_repository-1.1**. Because you can install all these packages using yum, no potential problem is detected. However, when the client installs the **example_tools-1.1** package, a dependency resolution problem occurs because both **example_repository-libs-1.0** and **example_repository-libs-1.1** cannot be installed.

There is currently no workaround for this problem. The larger the time frame, and major *Y* releases between the base set of RPMs and the errata being applied, the higher the chance of a problem with dependency resolution.

11.4. CREATING A CONTENT VIEW FILTER FOR ERRATA

You can use 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.

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"
```

11.5. ADDING ERRATA TO AN INCREMENTAL CONTENT VIEW

If errata are available but not installable, you can create an incremental Content View version to add the errata to your content hosts. For example, if the Content View is version 1.0, it becomes Content View version 1.1, and when you publish, it becomes Content View version 2.0.

1. In the Satellite web UI, navigate to **Content** > **Errata**.
2. From the **Errata** list, click the name of the errata that you want to apply.
3. Select the content hosts that you want to apply the errata to, and click **Apply to Hosts**. This creates the incremental update to the Content View.
4. If you want to apply the errata to the content host, select the **Apply Errata to Content Hosts immediately after publishing** check box.

   NOTE

   Until BZ#1459807 is resolved, if you apply non-installable errata to hosts registered to Capsule Servers, do not select the **Apply errata to Content Hosts immediately after publishing** check box.

   Instead, after clicking **Confirm**, wait for the errata Content View to be promoted and for the Capsule synchronization task to finish. Then, the errata will be marked as **Installable** and you can use the procedure again to apply it.

5. Click **Confirm** to apply the errata.

For CLI Users

1. List the errata and its corresponding IDs:

   ```
   # hammer erratum list
   ```

2. List the different content-view versions and the corresponding IDs:

   ```
   # hammer content-view version list
   ```

3. Apply a single erratum to content-view version. You can add more IDs in a comma-separated list.

   ```
   # hammer content-view version incremental-update \
   --content-view-version-id 319 --errata-ids 34068b
   ```

11.6. APPLYING ERRATA TO A HOST

Use these procedures to review and apply errata to a host.

Prerequisites

- Synchronize Red Hat Satellite repositories with the latest errata available from Red Hat. For more information, see Section 5.8, "Synchronizing Red Hat Repositories" .
- Register the host to an environment and Content View on Satellite Server. For more information, see Registering Hosts in the *Managing Hosts* guide.

- For RHEL 7 hosts, ensure that you install the **katello-agent** package. For more information, see Installing the Katello Agent in the *Managing Hosts* guide.

For Red Hat Enterprise Linux 8

To apply an erratum to a RHEL 8 host, you can run a remote execution job on Satellite Server or update the host. For more information about running remote execution jobs, see Running Jobs on Hosts in the *Managing Hosts* guide.

To apply an erratum to a RHEL 8 host, complete the following steps:

1. On Satellite, list all errata for the host:

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

2. Find the module stream an erratum belongs to:

```
# hammer erratum info --id ERRATUM_ID
```

3. On the host, update the module stream:

```
# yum update Module_Stream_Name
```

For Red Hat Enterprise Linux 7

To apply an erratum to a RHEL 7 host, complete the following steps:

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

To apply an erratum to a RHEL 7 host, complete the following steps:

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...
```

11.7. APPLYING ERRATA TO MULTIPLE HOSTS

Use these procedures to review and apply errata to multiple RHEL 7 hosts.

Prerequisites

- Synchronize Red Hat Satellite repositories with the latest errata available from Red Hat. For more information, see Section 5.8, “Synchronizing Red Hat Repositories” .
- Register the hosts to an environment and Content View on Satellite Server. For more information, see Registering Hosts in the *Managing Hosts* guide.
- Install the **katello-agent** package on hosts. For more information, see Installing the Katello Agent 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 hosts 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 hosts that this erratum is applicable to:

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

3. Apply the errata to a single host:

```
# 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 host 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 12. 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 6.5, 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
```

12.1. 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 repisotry set you want to use, for example, the **Red Hat Enterprise Linux Atomic Host Trees**set 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"
```

12.2. 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"
```

12.3. 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 13. 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.

13.1. 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"
```

13.2. 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 **Create 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 14. 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.

14.1. 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 14.1. Optional Parameters for the hammer product create Command

Option	Description
--gpg-key *gpg_key_name*	Key name to search by
--gpg-key-id *gpg_key_id*	GPG key numeric identifier
--sync-plan *sync_plan_name*	Sync plan name to search by
--sync-plan-id *sync_plan_id*	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 14.2. Optional Parameters for the hammer repository create Command

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

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

14.2. 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 14.3, "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-6.5-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.

14.3. 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 14.2, "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-6.5-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 14.1, “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.

14.4. 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.

14.5. 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 14.1, "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 15. 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.

15.1. 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"
```

15.2. 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 15.1, “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"
```

15.3. 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 **Create 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.

15.4. 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
```

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. Stop the **satellite-maintain** services on the Satellite host:

   ```
   # satellite-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-maintain** services on the Satellite host:

```
# satellite-maintain service start
```

Satellite Server now uses the NFS share to store content. Run a content synchronization (see Section 5.2, “Content Synchronization Overview”) to ensure the NFS share works as expected.

APPENDIX B. IMPORTING CONTENT ISO IMAGES 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.

B.1. IMPORTING CONTENT ISO IMAGES

Download the product ISO image from the Red Hat Customer Portal, as follows:

1. Log on to the Red Hat Customer Portal .
2. At the top of the screen, click **Downloads** and select **Red Hat Satellite**.
3. Click the link for the product name, such as Red Hat Enterprise Linux 6 Server (x86_64) to download the ISO image.
4. Copy all of Satellite Content ISO images to a directory Satellite can access. This example uses **/root/isos**.
5. Copy all of Satellite Content ISOs to a directory that Satellite can access. This example uses **/root/isos**.
6. Create a local directory that is shared through httpd on Satellite. This example uses **/var/www/html/pub/sat-import/**.

   ```
   # mkdir -p /var/www/html/pub/sat-import/
   ```

7. Mount and recursively copy the contents of the first ISO image 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
   ```

8. Repeat the above step for each ISO image until you have copied all the data from the Content ISO images into **/var/www/html/pub/sat-import/**.
9. Ensure the SELinux context for the directory is correct:

   ```
   # restorecon -rv /var/www/html/pub/sat-import/
   ```

10. Satellite Server now contains the content from the Content ISO images. However, Satellite Server needs to point to this location as the CDN URL. In the Satellite Web UI, navigate to **Content** > **Subscriptions**.
11. Click **Manage Manifest**.

12. Edit the **Red Hat CDN URL** field to point to the Satellite host name with the newly created directory, for example: **http://server.example.com/pub/sat-import/**

13. Click **Update** and then upload your manifest using Section 4.1, “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** > **Subscriptions**.

2. Click **Manage Manifest**.

3. Edit the **Red Hat CDN URL** field to point to the Red Hat CDN URL: **https://cdn.redhat.com**

4. Click **Save**

Satellite Server pulls content directly from Red Hat Customer Portal on the next synchronization.

B.2. IMPORTING KICKSTART REPOSITORIES

Kickstart repositories are not provided by the Content ISO image. To use Kickstart repositories in your disconnected Satellite, you must download a binary DVD ISO file for the version of Red Hat Enterprise Linux that you want to use and copy the Kickstart files to Satellite.

Procedure

1. Navigate to the Red Hat Customer Portal at https://access.redhat.com/ and log on.

2. In the upper left of the window, click **Downloads**.

3. Locate and click the version of Red Hat Enterprise Linux that you want to use, for example **Red Hat Enterprise Linux 8**.

4. In the Download Red Hat Enterprise Linux window, locate the binary DVD version of the ISO image, for example, **Red Hat Enterprise Linux 8.1 Binary DVD** and click **Download Now**.

5. When the download completes, copy the ISO image to Satellite Server.

6. On Satellite Server, create a mount point and temporarily mount the ISO image at that location:

```
# mkdir /mnt/iso
# mount -o loop rhel-8.1-x86_64-dvd.iso /mnt/iso
```

7. Create Kickstart directories for AppStream and BaseOS:

```
# mkdir /var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/appstream/kickstart

# mkdir /var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/baseos/kickstart
```

Note that if you use Red Hat Enterprise Linux 7, you must create and complete all the following steps in only one directory **/var/www/html/pub/sat-import/content/dist/rhel/server/7/7.7/x86_64/kickstart/**.

8. To the listing files **/var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/appstream/listing** and **/var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/baseos/listing**, append **kickstart** with a new line:

   ```
   kickstart
   ```

9. To the listing file **/var/www/html/pub/sat-import/content/dist/rhel8/listing**, append the version number of the operating system ISO that you use with a new line. For example, for the RHEL 8.1 binary ISO, add **8.1** with a new line:

   ```
   8.1
   ```

10. Copy the **kickstart** files from the ISO image:

    ```
    # cp -a /mnt/iso/AppStream/* \
    /var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/appstream/kickstart

    # cp -a /mnt/iso/BaseOS/* /mnt/iso/images/ \
    /var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/baseos/kickstart
    ```

    Note that for BaseOS, you must also copy the contents of the **/mnt/*iso*/images/** directory.

11. Copy the **.treeinfo** files from the ISO image:

    ```
    # cp /mnt/iso/.treeinfo \
    /var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/appstream/kickstart/treeinfo

    # cp /mnt/iso/.treeinfo \
    /var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/baseos/kickstart/treeinfo
    ```

12. Open the **/var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/baseos/kickstart/treeinfo** file for editing.

13. In the **[general]** section, make the following changes:

    - Change **packagedir = AppStream/Packages** to **packagedir = Packages**
    - Change **repository = AppStream** to **repository = .**
    - Change **variant = AppStream** to **variant = BaseOS**
    - Change **variants = AppStream,BaseOS** to **variants = BaseOS**

14. In the **[tree]** section, change **variants = AppStream,BaseOS** to **variants = BaseOS**.

15. In the **[variant-BaseOS]** section, make the following changes:

    - Change **packages = BaseOS/Packages** to **packages = Packages**
    - Change **repository = BaseOS** to **repository = .**

16. Delete the **[media]** and **[variant-AppStream]** sections.

17. Save and close the file.

18. Verify that the **/var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/baseos/kickstart/treeinfo** file has the following format:

```
[checksums]
images/efiboot.img =
sha256:9ad9beee4c906cd05d227a1be7a499c8d2f20b3891c79831325844c845262bb6
images/install.img =
sha256:e246bf4aedfff3bb54ae9012f959597cdab7387aadb3a504f841bdc2c35fe75e
images/pxeboot/initrd.img =
sha256:a66e3c158f02840b19c372136a522177a2ab4bd91cb7269fb5bfdaaf7452efef
images/pxeboot/vmlinuz =
sha256:789028335b64ddad343f61f2abfdc9819ed8e9dfad4df43a2694c0a0ba780d16

[general]
; WARNING.0 = This section provides compatibility with pre-productmd treeinfos.
; WARNING.1 = Read productmd documentation for details about new format.
arch = x86_64
family = Red Hat Enterprise Linux
name = Red Hat Enterprise Linux 8.1.0
packagedir = Packages
platforms = x86_64,xen
repository = .
timestamp = 1571146127
variant = BaseOS
variants = BaseOS
version = 8.1.0

[header]
type = productmd.treeinfo
version = 1.2

[images-x86_64]
efiboot.img = images/efiboot.img
initrd = images/pxeboot/initrd.img
kernel = images/pxeboot/vmlinuz

[images-xen]
initrd = images/pxeboot/initrd.img
kernel = images/pxeboot/vmlinuz

[release]
name = Red Hat Enterprise Linux
short = RHEL
version = 8.1.0

[stage2]
mainimage = images/install.img

[tree]
arch = x86_64
build_timestamp = 1571146127
platforms = x86_64,xen
variants = BaseOS
```

```
[variant-BaseOS]
id = BaseOS
name = BaseOS
packages = Packages
repository = .
type = variant
uid = BaseOS
```

19. Open the **/var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/appstream/kickstart/treeinfo** file for editing.

20. In the **[general]** section, make the following changes:

    - Change **packagedir = AppStream/Packages** to **packagedir = Packages**
    - Change **repository = AppStream** to **repository = .**
    - Change **variants = AppStream,BaseOS** to **variants = AppStream**

21. In the **[tree]** section, change **variants = AppStream,BaseOS** to **variants = AppStream**.

22. In the **[variant-AppStream]** section, make the following changes:

    - Change **packages = AppStream/Packages** to **packages = Packages**
    - Change **repository = AppStream** to **repository = .**

23. Delete the following sections from the file: **[checksums]**, **[images-x86_64]**, **[images-xen]**, **[media]**, **[stage2]**, **[variant-BaseOS]**.

24. Save and close the file.

25. Verify that the **/var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/appstream/kickstart/treeinfo** file has the following format:

```
[general]
; WARNING.0 = This section provides compatibility with pre-productmd treeinfos.
; WARNING.1 = Read productmd documentation for details about new format.
arch = x86_64
family = Red Hat Enterprise Linux
name = Red Hat Enterprise Linux 8.1.0
packagedir = Packages
platforms = x86_64,xen
repository = .
timestamp = 1571146127
variant = AppStream
variants = AppStream
version = 8.1.0

[header]
type = productmd.treeinfo
version = 1.2

[release]
name = Red Hat Enterprise Linux
short = RHEL
```

```
version = 8.1.0

[tree]
arch = x86_64
build_timestamp = 1571146127
platforms = x86_64,xen
variants = AppStream

[variant-AppStream]
id = AppStream
name = AppStream
packages = Packages
repository = .
type = variant
uid = AppStream
```

26. If you do not plan to use the mounted binary DVD ISO image, unmount and remove the directory:

    ```
    # umount /mnt/iso
    # rmdir /mnt/iso
    ```

27. In the Satellite web UI, enable the Kickstart repositories. For more information, see Section 5.7, “Enabling Red Hat Repositories”.

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.

IMPORTANT

You can only import content ISO images for Red Hat Enterprise Linux 8 because repodata checksum from CDN does not match the repodata checksum from the content ISO images for Red Hat Enterprise Linux 7 and lower.

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 procedure performs the first synchronization of the Red Hat Enterprise Linux 8 repository from content ISO images.

Procedure

1. Log in to the Red Hat Customer Portal at https://access.redhat.com/.
2. In the upper left of the window, click **Downloads** and select **Red Hat Satellite**.
3. Click the Content ISOs tab. This page lists all the products that are available in your subscription.
4. Click the link for the product name, such as RHEL 8 (x86_64), to reveal links to download ISO images.
5. Download the ISO images.
6. On Satellite Server, create a directory to act as a temporary store for all of the required Satellite content ISO images. This example uses **/tmp/isos/rhel8**:

```
# mkdir -p /tmp/isos/rhel8
```

7. On your workstation, copy the ISO files to Satellite Server:

```
$ scp ~/Downloads/iso_file root@satellite.example.com:/tmp/isos/rhel8
```

8. On Satellite Server, create a directory to serve as a mount point for the ISOs:

```
# mkdir /mnt/iso
```

9. Create a working directory to store ISO images:

```
# mkdir /mnt/rhel8
```

10. Temporarily mount the first ISO image:

```
# mount -o loop /tmp/isos/iso_file /mnt/iso
```

11. Recursively copy the contents of the first ISO to the working directory:

```
# cp -ruv /mnt/iso/* /mnt/rhel8/
```

12. Unmount the ISO image:

```
# umount /mnt/iso
```

13. Repeat the above step for each ISO until you have copied all the data from the Content ISO images into **/mnt/rhel8**.

14. If required, remove the empty directory used as the mount point:

```
# rmdir /mnt/iso
```

15. If required, remove the temporary working directory and its contents to regain space:

```
# rm -rf /tmp/isos/
```

16. 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/rhel8/
# chown -R apache:apache /mnt/rhel8/
```

17. Create or edit the **/etc/pulp/content/sources/conf.d/local.conf** file. Append the following text into the file:

```
[rhel-8-server]
enabled: 1
priority: 0
expires: 3d
name: Red Hat Enterprise Linux 8
type: yum
base_url: file:///mnt/rhel8/content/dist/rhel/server/8/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**.

18. In the Satellite web UI, navigate to **Content** > **Red Hat Repositories** and enable the following repositories:

- **Red Hat Enterprise Linux 8 for x86_64 - BaseOS RPMs 8**

- **Red Hat Enterprise Linux 8 for x86_64 - AppStream RPMs 8**

19. Under **Content** > **Sync Status** select the repositories to be synchronized and click **Synchronize Now**.

Note that the Satellite web UI does not indicate 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 on Satellite:

```
# 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.

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	**associate** Accepted values: **always**, **new**, **never**	Associates templates with OS, Organization, and Location based on metadata.	N/A
Branch	**branch**	Specifies the default branch in Git repository to read from.	Specifies the default branch in Git repository to write to.
Dirname	**dirname**	Specifies the subdirectory under the repository to read from.	Specifies the subdirectory under the repository to write to.
Filter	**filter**	Imports only templates with names that match this regular expression.	Exports only templates with names that match this regular expression.
Force import	**force**	Imported templates overwrite locked templates with the same name.	N/A

Parameter	API parameter name	Meaning on importing	Meaning on exporting
Metadata export mode	**metadata_export_mode** Accepted values: **refresh**, **keep**, **remove**	N/A	Defines how metadata is handled when exporting: ● **Refresh** – remove existing metadata from the template content and generate new metadata based on current assignments and attributes. ● **Keep** – retain the existing metadata. ● **Remove** – export template without metadata. Useful if you want to add metadata manually.
Negate	**negate** Accepted values: **true**, **false**	Imports templates ignoring the filter attribute.	Exports templates ignoring the filter attribute.
Prefix	**prefix**	Adds specified string to the beginning of the template if the template name does not start with the prefix already.	N/A
Repo	**repo**	Defines the path to the repository to synchronize from.	Defines the path to a repository to export to.
Verbosity	**verbose** Accepted values: **true**, **false**	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 authentification 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
```