Red Hat Satellite 6.10

Content Management Guide

A guide to managing content from Red Hat and custom sources

Last Updated: 2023-06-21

Red Hat Satellite 6.10 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, and ISO images. Red Hat Satellite 6 manages this content using a set of Content Views promoted across the application lifecycle. This guide demonstrates how to create an application lifecycle that suits your organization and content views that fulfils host states within lifecycle environments. These content views eventually form the basis for provisioning and updating hosts in your Red Hat Satellite 6 environment.

Table of Contents

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 other types of custom content in Satellite. 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.

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. To use the CLI instead of the web UI, see CLI procedure.

Procedure

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**.

CLI procedure

1. To create an organization, enter the following command:

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

2. Optional: To edit an organization, enter the **hammer organization update** command. For example, the following command assigns a compute resource to the organization:

```
# hammer organization update \
--name "your_organization_name" \
--compute-resource-ids 1
```

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.

CLI procedure

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/content
```

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/content/My_Organization_Label/Library/content/dist/rhel/server/7/7Server/x86_64/sat-tools/6.10/os/
```

Ensure that the paths to **cert.pem** and **key.pem** are the correct absolute paths otherwise the command fails silently. Pulp uses the organization label, therefore, you must enter the organization label into the URL.

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

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.

CLI procedure

1. Enter the following command to retrieve the ID of the organization that you want to delete:

   ```
   # hammer organization list
   ```

   From the output, note the ID of the organization that you want to delete.

2. Enter the following command to delete an organization:

   ```
   # hammer organization delete --id Organization_ID
   ```

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. To use the CLI instead of the web UI, see the CLI procedure.

Procedure

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.

CLI procedure

- 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. CREATING MULTIPLE LOCATIONS

The following example Bash script creates three locations - London, Munich, Boston - and assigns them to the Example Organization.

```
ORG="Example Organization"
LOCATIONS="London Munich Boston"
```

```
for LOC in ${LOCATIONS}
do
  hammer location create --name "${LOC}"
  hammer location add-organization --name "${LOC}" --organization "${ORG}"
done
```

3.3. 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.

CLI procedure

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*.

3.4. 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.

CLI procedure

1. Enter the following command to retrieve the ID of the location that you want to delete:

   ```
   # hammer location list
   ```

   From the output, note the ID of the location that you want to delete.

2. Enter the following command to delete the location:

```
# hammer location delete --id Location ID
```

CHAPTER 4. MANAGING SUBSCRIPTIONS

Red Hat Satellite 6 can import 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.

Prerequisites

- 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

1. In the Satellite web UI, ensure the context is set to the organization you want to use.
2. Navigate to **Content** > **Subscriptions** and click **Manage Manifest**.
3. In the Manage Manifest window, click **Browse**.
4. Navigate to the location that contains the Subscription Manifest file, then click **Open**. If the Manage Manifest window does not close automatically, click **Close** to return to the Subscriptions window.

CLI procedure

1. Copy the Subscription Manifest file from your client to Satellite Server:

```
$ scp ~/manifest_file.zip root@satellite.example.com:~/.
```

2. Log in to Satellite Server as the **root** user and import the Subscription Manifest file:

```
# 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 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” .

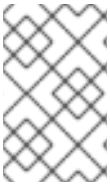
Procedure

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 9, *Managing Activation Keys*.

Satellite Subscriptions

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

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

Prerequisite

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

Procedure

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.

CLI procedure

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.

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.

CHAPTER 5. IMPORTING CONTENT

This chapter outlines how you can import different types of custom content to Satellite. If you want to import Rpms, Files, 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 13, *Managing ISO Images*
- Chapter 14, *Managing Custom File Type Content*

5.1. USING PRODUCTS AND REPOSITORIES IN SATELLITE

Both Red Hat content and custom content in Satellite 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. Satellite creates a subscription for each custom product you create.

Red Hat Content is already organized into Products. For example, Red Hat Enterprise Linux Server is a *Product* in Satellite. The repositories for that Product consist of different versions, architectures, and add-ons. For Red Hat repositories, products are created automatically after enabling the repository. See Section 5.5, “Enabling Red Hat Repositories”

Other custom content can be organized into Products however you want. For example, you might create an EPEL (Extra Packages for Enterprise Linux) product and add an "EPEL 7 x86_64" repository to it.

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

5.2. IMPORTING CUSTOM SSL CERTIFICATES

Before you synchronize custom content from an external source, you might need to import SSL certificates into your custom product. This might include client certs and keys or CA certificates for the upstream repositories you want to synchronize.

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**.

5.3. CREATING A CUSTOM PRODUCT

Use this procedure to create a custom product that you can then add repositories to. To use the CLI instead of the web UI, see the CLI procedure.

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. Satellite 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**.

CLI procedure

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"
```

5.4. ADDING CUSTOM RPM REPOSITORIES

Use this procedure to add custom RPM repositories in Satellite. To use the CLI instead of the web UI, see the CLI procedure.

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 **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 **New 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**), Docker images (**docker**), Files (**file**) or other. Note that if the **yum** content type is selected, you can restrict whether the repository is available to a host based on the host's architecture and OS version.

4. Optional: From the **Restrict to Architecture** list, select the architecture. Ensure that **No restriction**, which is the default value, is selected to make the repository available to all hosts regardless of the architecture.

5. Optional: From the **Restrict to OS Version** list, select the OS version. Ensure that **No restriction**, which is the default value, is selected to make the repository available to all hosts regardless of the OS version.

6. In the **URL** field, enter the URL of the external repository to use as a source. Satellite supports three protocols: **http://**, **https://**, and **file://**. If you are using a **file://** repository, you have to place it under **/var/lib/pulp/sync_imports/** directory.

7. From the **Download Policy** list, select the type of synchronization Satellite Server performs. See Section 5.8, "Download Policies Overview"

8. 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.

9. From the **Checksum** list, select the checksum type for the repository.

10. Optional: If you want, you can clear the **Publish via HTTP** check box to disable this repository from publishing through HTTP.

11. Optional: From the **GPG Key** list, select the GPG key for the product.

12. Click **Save**.

CLI procedure

1. Enter the following command to create the repository:

```
# hammer repository create \
--name "My_Repository" \
--content-type "yum" \
--os-version "My_OS_Version" \
--arch "My_System_Architecture" \
--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"
```

Continue to Section 5.6, "Syncing Repositories" to sync the repository

5.5. ENABLING RED HAT REPOSITORIES

If outside network access requires usage of an HTTP Proxy, configure a default HTTP Proxy for your server. See Adding a default HTTP Proxy

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 configure Satellite to synchronize content with a local CDN server before synchronizing content. For more information, see Appendix B, *Configuring Satellite to Synchronize Content with a Local CDN Server*.

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 7 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.

CLI procedure

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"
```

5.6. SYNCING REPOSITORIES

Procedure

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.

CLI procedure

- Synchronize an entire Product:

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

- Synchronize the repository individually:

```
# hammer repository synchronize \
--name "My_Repository" \
--product "My Product" \
--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. See: Section 5.16, “Creating a Synchronization Plan”

5.7. 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
```

5.8. 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.

The **On Demand** policy acts 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.

Inherit

Capsule Server inherits the download policy for the repository from the corresponding repository on Satellite Server.

5.9. CHANGING THE DEFAULT DOWNLOAD POLICY

You can set the default download policy that Satellite applies to repositories that you create in all organizations.

Depending on whether it is a Red Hat or non-Red Hat custom repository, Satellite uses separate settings. 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.
3. Change the default download policy depending on your requirements:
   - To change the default download policy for a Red Hat repository, change the value of the **Default Red Hat Repository download policy**setting.
   - To change the default download policy for a custom repository, change the value of the **Default Custom Repository download policy**setting.

CLI procedure

- To change the default download policy for Red Hat repositories to one of **immediate** or **on_demand**, enter the following command:

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

- To change the default download policy for a non-Red Hat custom repository to one of **immediate** or **on_demand**, enter the following command:

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

5.10. CHANGING THE DOWNLOAD POLICY FOR A REPOSITORY

You can set the download policy for a repository.

Procedure

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**.

CLI procedure

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** or **on_demand**:

```
# 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
```

5.11. UPLOADING CONTENT TO CUSTOM RPM REPOSITORIES

You can upload individual RPMs and source RPMs to custom RPM repositories. You can upload RPMs using the Satellite web UI or the Hammer CLI. You must use the Hammer CLI to upload source RPMs.

Procedure

1. In the Satellite web UI, click **Content** > **Products**.
2. Click the name of the custom product.
3. In the **Repositories** tab, click the name of the custom RPM repository.
4. Under **Upload Package**, click **Browse...** and select the RPM you want to upload.
5. Click **Upload**.

To view all RPMs in this repository, click the number next to **Packages** under **Content Counts**.

CLI procedure

- Enter the following command to upload an RPM:

```
# hammer repository upload-content \
--id repo_ID \
--path /path/to/example-package.rpm
```

- Enter the following command to upload a source RPM:

```
# hammer repository upload-content \
--content-type srpm \
--id repo_ID \
--path /path/to/example-package.src.rpm
```

When the upload is complete, you can view information about a source RPM by using the commands **hammer srpm list** and **hammer srpm info --id *srpm_ID***.

5.12. 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.

Verify Content Checksum

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

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. To perform optimized sync or complete sync, select **Advanced Sync** from the **Select Action** menu.
5. Select the required option and click **Sync**.
6. To verify the checksum, click **Verify Content Checksum** from the **Select Action** menu. (optional)

CLI procedure

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 --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
```

5.13. ADDING A NEW HTTP PROXY

Use this procedure to add HTTP proxies to Satellite. You can then specify which HTTP proxy to use for Products, repositories, and supported compute resources.

If Satellite Server uses a proxy to communicate with subscription.rhsm.redhat.com or subscription.rhn.redhat.com, and cdn.redhat.com then the proxy must not perform SSL inspection on these communications.

To use the CLI instead of the web UI, see the CLI procedure.

Procedure

1. In the Satellite web UI, navigate to **Infrastructure** > **HTTP Proxies** and select **New HTTP Proxy**.
2. In the **Name** field, enter a name for the HTTP proxy.
3. In the **URL** field, enter the URL for the HTTP proxy, including the port number. The following host names are available:

Host name	Port	Protocol
subscription.rhsm.redhat.com	443	HTTPS
subscription.rhn.redhat.com	443	HTTPS
cdn.redhat.com	443	HTTPS
api.access.redhat.com (if using Red Hat Insights)	443	HTTPS
cert-api.access.redhat.com (if using Red Hat Insights)	443	HTTPS

4. If your HTTP proxy requires authentication, enter a **Username** and **Password**.
5. Optional: In the **Test URL** field, enter the HTTP proxy URL, then click **Test Connection** to ensure that you can connect to the HTTP proxy from Satellite.
6. Click the **Locations** tab and add a location.
7. Click the **Organization** tab and add an organization.
8. Click **Submit**.

CLI procedure

- On Satellite Server, enter the following command to add a new HTTP proxy:

```
# hammer http-proxy create --name proxy-name \
--url proxy-URL:port-number
```

-

If your HTTP proxy requires authentication, add the **--username *name*** and **--password *password*** options.

For further information, see the Knowledgebase article How to access Red Hat Subscription Manager (RHSM) through a firewall or proxy on the Red Hat Customer Portal.

5.14. CHANGING THE HTTP PROXY POLICY FOR A PRODUCT

For granular control over network traffic, you can set an HTTP proxy policy for each Product. A Product's HTTP proxy policy applies to all repositories in the Product, unless you set a different policy for individual repositories.

To set an HTTP proxy policy for individual repositories, see Section 5.15, "Changing the HTTP Proxy Policy for a Repository".

Procedure

1. In the Satellite web UI, navigate to **Content** > **Products** and select the check box next to each of the Products that you want to change.
2. From the **Select Action** list, select **Manage HTTP Proxy**.
3. Select an **HTTP Proxy Policy** from the list:
   - **Global Default**: Use the global default proxy setting.
   - **No HTTP Proxy**: Do not use an HTTP proxy, even if a global default proxy is configured.
   - **Use specific HTTP Proxy**: Select an **HTTP Proxy** from the list. You must add HTTP proxies to Satellite before you can select a proxy from this list. For more information, see Section 5.13, "Adding a New HTTP Proxy" .
4. Click **Update**.

5.15. CHANGING THE HTTP PROXY POLICY FOR A REPOSITORY

For granular control over network traffic, you can set an HTTP proxy policy for each repository. To use the CLI instead of the web UI, see the CLI procedure.

To set the same HTTP proxy policy for all repositories in a Product, see Section 5.14, "Changing the HTTP Proxy Policy for a Product".

Procedure

1. In the Satellite web UI, navigate to **Content** > **Products** and click the name of the Product that contains the repository.
2. In the **Repositories** tab, click the name of the repository.
3. Locate the **HTTP Proxy** field and click the edit icon.
4. Select an **HTTP Proxy Policy** from the list:
   - **Global Default**: Use the global default proxy setting.

- **No HTTP Proxy**: Do not use an HTTP proxy, even if a global default proxy is configured.
- **Use specific HTTP Proxy**: Select an **HTTP Proxy** from the list. You must add HTTP proxies to Satellite before you can select a proxy from this list. For more information, see Section 5.13, “Adding a New HTTP Proxy” .

5. Click **Save**.

CLI procedure

- On Satellite Server, enter the following command, specifying the HTTP proxy policy you want to use:

```
# hammer repository update --id repository-ID \
--http-proxy-policy policy
```

Specify one of the following options for **--http-proxy-policy**:

- **none**: Do not use an HTTP proxy, even if a global default proxy is configured.
- **global_default_http_proxy**: Use the global default proxy setting.
- **use_selected_http_proxy**: Specify an HTTP proxy using either **--http-proxy *proxy-name*** or **--http-proxy-id *proxy-ID***. To add a new HTTP proxy to Satellite, see Section 5.13, “Adding a New HTTP Proxy”.

5.16. 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.

To use the CLI instead of the web UI, see the CLI procedure.

Procedure

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**.

CLI procedure

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"
```

3. View the available synchronization plans for an organization to verify that the synchronization plan is created:

```
# hammer sync-plan list --organization "Default Organization"
```

5.17. ASSIGNING A SYNCHRONIZATION PLAN TO MULTIPLE PRODUCTS

Use this procedure to assign a synchronization plan to the products in an organization that have been synchronized at least once and contain at least one repository

Procedure

To assign a synchronization plan to the selected products, complete the following steps:

1. Run the following Bash script:

```
ORG="My_Organization"
SYNC_PLAN="daily_sync_at_3_a.m"

hammer sync-plan create --name $SYNC_PLAN --interval daily --sync-date "2023-04-5
03:00:00" --enabled true --organization $ORG
for i in $(hammer --no-headers --csv --csv-separator="|" product list --organization $ORG --
per-page 999 | grep -vi not_synced | awk -F'|' '$5 != "0" { print $1}')
do
  hammer product set-sync-plan --sync-plan $SYNC_PLAN --organization $ORG --id $i
done
```

2. After executing the script, view the products assigned the synchronization plan:

```
# hammer product list --organization $ORG --sync-plan $SYNC_PLAN
```

5.18. LIMITING SYNCHRONIZATION CONCURRENCY

By default each Repository Synchronization job can fetch up to 10 files at a time. This can be adjusted on a per repository basis.

Increasing the limit may improve performance, but can cause the upstream server to be overloaded or start rejecting requests. If you are seeing Repository syncs fail due to the upstream servers rejecting requests, you may want to try lowering the limit.

To do so from the CLI:

```
# hammer repository update --organization $ORG --download-concurrency 5 --id $REPO_ID
```

5.19. IMPORTING A CUSTOM GPG KEY

When clients are consuming signed custom content, ensure that the clients are configured to validate the installation of RPMs with the appropriate GPG Key. This helps to ensure that only packages from authorized sources can be installed.

Red Hat content is already configured with the appropriate GPG key and thus GPG Key management of Red Hat Repositories is not supported.

To use the CLI instead of the web UI, see the CLI procedure.

Prerequisites

Ensure that you have a copy of the GPG key used to sign the RPM content that you want to use and manage in Satellite. Most RPM distribution providers provide their GPG Key on their website. You can also extract this manually from an RPM:

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

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
Wl5GnzcLGAnUSRamfqGUWcyMMinHHIKIc1X1P4I=
=WPpI
-----END PGP PUBLIC KEY BLOCK-----
```

4. Click **Save**.

CLI procedure

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"
```

CHAPTER 6. 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.

6.1. 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

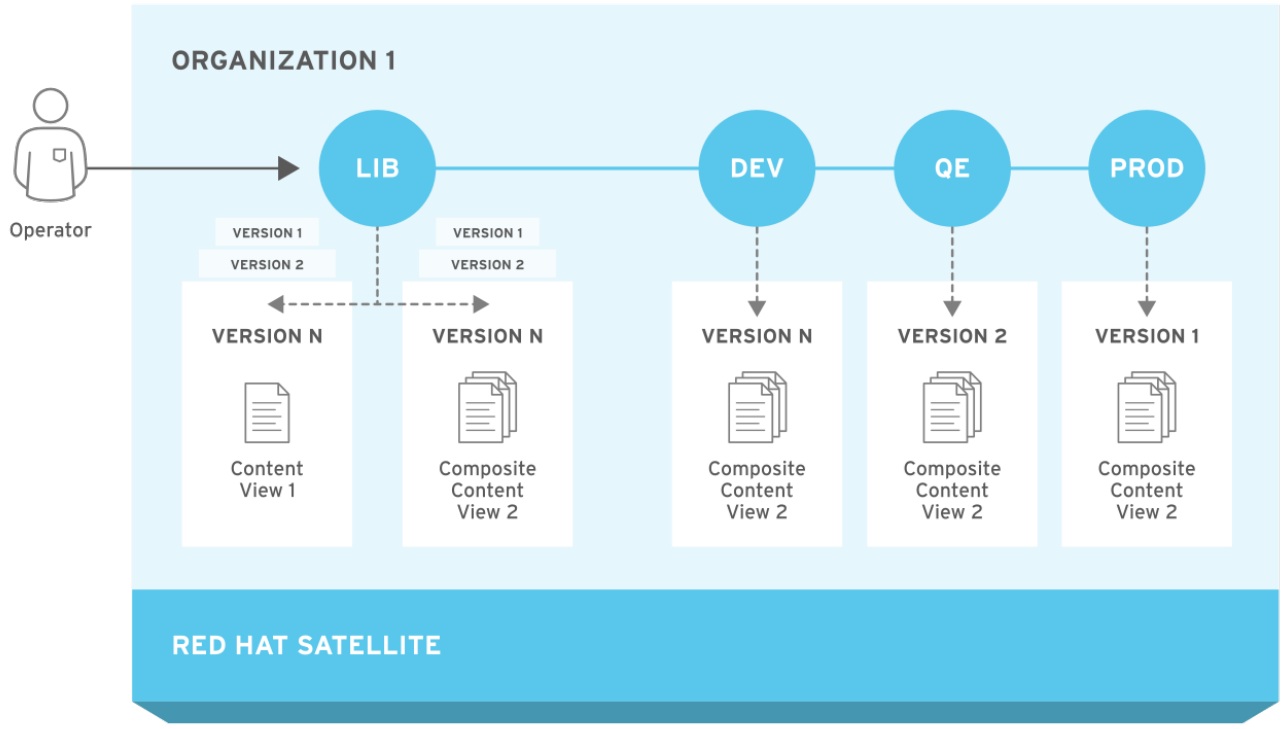
Red Hat Satellite 6 provides methods to customize each application life cycle stage so that it suits your specifications.

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, and packages 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 6.1. The Red Hat Satellite 6 Application Life Cycle



6.2. 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 7.3, “Promoting a Content View” .

6.3. 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.

CLI procedure

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"
```

6.4. REMOVING LIFE CYCLE ENVIRONMENTS FROM SATELLITE SERVER

Use this procedure to remove a life cycle environment.

Procedure

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.

CLI procedure

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"
```

6.5. 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

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**.

CLI procedure

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

```
# hammer capsule list
```

2. To verify 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 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 \
--lifecycle-environment-id lifecycle_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
```

6.6. 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

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 click **Submit**.
4. To synchronize the content on the Capsule, click the **Overview** tab and click **Synchronize**.
5. Select either **Optimized Sync** or **Complete Sync**.
   For definitions of each synchronization type, see Recovering a Repository.

CLI procedure

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

   ```
   # hammer capsule list
   ```

   Note the Capsule ID of the Capsule that you want to add a life cycle to.
2. Using the ID, verify the details of your Capsule:

   ```
   # hammer capsule info --id capsule_id
   ```
3. 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
   ```
4. Add the life cycle environment to your Capsule Server:

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

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

5. Synchronize the content from Satellite to Capsule.

   - 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 \
     --lifecycle-environment-id lifecycle-environment_id
     ```

CHAPTER 7. 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, errata filters, and module stream 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 one or more repositories that you want to the Content View.
3. Optionally, create one or more filters to refine the content of the Content View.
4. Optionally, resolve any package dependencies for a Content View.
5. Publish the Content View.
6. Optionally, promote the Content View to another environment.
7. 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 7.6, "Creating a Composite Content View".

Package Dependency Resolution

Package dependency is a complication of package management. For more information about how to manage package dependencies within Content Views, see Section 7.8, "Resolving Package Dependencies".

7.1. CREATING A CONTENT VIEW

Use this procedure to create a simple Content View. To use the CLI instead of the web UI, see the CLI procedure.

Prerequisites

While you can stipulate whether you want to resolve any package dependencies on a Content View by Content View basis, you might want to change the default Satellite settings to enable or disable package resolution for all Content Views. For more information, see Section 7.8, "Resolving Package Dependencies".

Procedure

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. Satellite automatically completes the **Label** field from the name you enter.
3. In the **Description** field, enter a description of the view.
4. Optional: if you want to solve dependencies automatically every time you publish this Content View, select the **Solve Dependencies** check box. Dependency solving slows the publishing time and might ignore any Content View filters you use. This can also cause errors when resolving dependencies for errata.
5. Click **Save** to create the Content View.
6. In the **Repository Selection** area, select the repositories that you want to add to your Content View, then click **Add Repositories**.
7. Click **Publish New Version** and in the **Description** field, enter information about the version to log changes.
8. Click **Save**.
9. 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.

CLI procedure

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"
```

4. Optional: To add a repository to an existing Content View, enter the following command:

```
# hammer content-view add-repository \
--name "Example_Content_View" \
--organization "My_Organization" \
--repository-id repository_ID
```

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

7.2. 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.

7.3. PROMOTING A CONTENT VIEW

Use this procedure to promote Content Views across different lifecycle environments. To use the CLI instead of the web UI, see the CLI procedure.

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

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.

CLI procedure

- 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.

7.4. PROMOTING A CONTENT VIEW ACROSS ALL LIFE CYCLE ENVIRONMENTS WITHIN AN ORGANIZATION

Use this procedure to promote Content Views across all lifecycle environments within an organization.

Procedure

To promote a Content Views version across all lifecycle environments within an organization, complete the following steps:

1. To promote a selected Content View version from Library across all life cycle environments within an organization, run the following Bash script:

```
ORG="Your_Organization"
CVV_ID=3

for i in $(hammer --no-headers --csv lifecycle-environment list --organization $ORG | awk -F, {'print $1'} | sort -n)
do
  hammer content-view version promote --organization $ORG --to-lifecycle-environment-id $i --id $CVV_ID
done
```

2. Display information about your Content View version to verify that it is promoted to the required lifecycle environments:

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

7.5. 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

example_software
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.

7.6. CREATING A COMPOSITE CONTENT VIEW

Use this procedure to create a composite content view. To use the CLI instead of the web UI, see the CLI procedure.

Procedure

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**.

CLI procedure

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"
```

7.7. 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 7.1. 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 blocklisted 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 five types of content to filter:

Table 7.2. Content Types

Content Type	Description
Package	Filter packages based on their name and version number. The **Package** option filters non-modular RPM packages and errata.
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.
Module Streams	Select whether to include or exclude specific module streams. The **Module Streams** option filters modular RPMs and errata, but does not filter non-modular content that is associated with the selected module stream.

7.8. RESOLVING PACKAGE DEPENDENCIES

In Satellite, you can use the package dependency resolution feature to ensure that any dependencies that packages have within a Content View are added to the dependent repository as part of the Content View publication process.

You can select to resolve package dependencies for any Content View that you want, or you can change the default setting to enable or disable resolving package dependencies for all new Content Views.

In most simple and common cases, dependency resolution is not necessary to the the correct function of the repository, and can be omitted. Note that resolving package dependencies can be a very expensive operation that can cause significant delays to Content View promotion. The package dependency resolution feature also does not consider packages that are installed on your system independently of the Content View, so even with dependency solving enabled, it is possible to use Content View filters which produce a repository for which **satellite-maintain packages update** is not guaranteed to work. Therefore, caution is recommended when using complex filter combinations, even in conjunction with dependency resolution.

Resolving Package Dependencies and Filters

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

If you add a filter that excludes some packages that are required and the Content View has dependency resolution enabled, Satellite ignores the rules you create in your filter in favor of resolving the package dependency.

If you create a content filter for security purposes, to resolve a package dependency, Satellite can add packages that you might consider insecure.

Procedure

To resolve package dependencies by default, complete the following steps:

1. In the Satellite web UI, navigate to **Administer** > **Settings** and click the **Content** tab.
2. Locate the **Content View Dependency Solving Default**, and select **Yes**.

Considerations

1. It is possible to use filters in such a way that the resulting Content View has broken dependencies, and dependency resolution is not always guaranteed to solve this. This is because the package dependency resolution feature does not consider packages that are installed on your system independently of the Content View.
2. Packages in a Content View can have conflicting dependencies.
   For example, imagine a scenario where A, B and C are example packages, where package A has a dependency on version 1.1 of package B, and package C has a dependency on version 1.2 of package B. B (1.1) and B (1.2) are both brought in by dependency solving in the Content View to satisfy dependencies of both A and C.
3. If a package in a repository has a dependency on a package outside the repository which has not been synced or added to the Content View, dependency solving will not solve dependencies for that package.
4. If you are using exclude filters on your Content View with dependency solving turned on, dependency solving takes precedence and brings in any required packages even if they were filtered out.
5. Dependency solving and filters can increase publishing times considerably.

7.9. 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.

Example 4

Filter a specific module stream in a Content View.

Filter 1:

- **Inclusion Type:** Include
- **Content Type:** Module Stream
- **Filter:** Select only the specific module stream that you want for the Content View, for example **ant**, and click **Add Module Stream**.

Filter 2:

- **Inclusion Type:** Exclude
- **Content Type:** Package

- **Filter:** Add a rule to filter any non-modular packages that you want to exclude from the Content View. If you do not filter the packages, the Content View filter includes all non-modular packages associated with the module stream **ant**. Add a rule to exclude all * packages, or specify the package names that you want to exclude.

For another example of how content filters work, see the following article: "How do content filters work in Satellite 6".

7.10. CREATING A CONTENT FILTER

Use this procedure to create a content filter. To use the CLI instead of the web UI, see the CLI procedure.

For examples of how to build a filter, see Section 7.9, "Content Filter Examples".

Procedure

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.

CLI procedure

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 8. SYNCHRONIZING CONTENT BETWEEN SATELLITE SERVERS

Red Hat Satellite uses Inter-Satellite Synchronization (ISS) to synchronize content between two Satellite Servers including those that are air-gapped.

Satellite 6.10 and 6.9 (and below) are incompatible for export and import due to different Pulp versions. You must be running the same version of Satellite on both servers to use this feature.

8.1. SCENARIOS

- 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 consumes content from Satellite Server, and you want to copy content from those Content Views to other Satellite Servers.
- If you want to copy all library content from your Satellite Server to another Satellite Servers. For example, you have Products and repositories that your IT department consumes content from Satellite Server in the Library, and you want to copy all Products and repositories in that organization to another Satellite Servers.
- 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 your management infrastructure for security or other purposes.

These scenarios require at least two Satellite Servers: one Satellite Server that is connected to the outside world and contains content to export, and another Satellite Server that is disconnected that you want to import content to.

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*.

8.2. CONFIGURATION DEFINITIONS

There are different ways of using ISS to synchronize content between two Satellite Servers.

Connected

In a connected scenario, two Satellite Servers are able to communicate with each other.

Disconnected

In a disconnected scenario, there is the following setup:

- Two Satellite Servers are connected to each other.
- One Satellite Server is connected to the Internet while the other Satellite Server is not connected to the Internet.
  - This is referred to as the disconnected Satellite Server.
- The disconnected Satellite Server is completely isolated from all external networks but can communicate with a connected Satellite Server.

Air-gapped Disconnected

In this setup, both Satellite Servers are isolated from each other. The only way for air-gapped Satellite Servers to receive content updates is through the import/export workflow.

8.3. EXPORTING FROM A CONNECTED SATELLITE SERVER

8.3.1. Connected Satellite Server as a Content Store

In this scenario, you have one connected Satellite Server that is receiving content from an external source like the CDN. The rest of your infrastructure is completely isolated, including another disconnected Satellite Server. The connected Satellite Server is mainly used as a content store for updates. The disconnected Satellite Server serves as the main Satellite Server for managing content for all infrastructure behind the isolated network.

On the connected Satellite Server

1. Ensure that repositories are using the immediate download policy, either by:
   a. For existing repos using **On Demand**, change their download policy on the repository details page to **Immediate**.
   b. For new repositories, ensure that the **Default Red Hat Repository download policy**setting is set to **Immediate** before enabling Red Hat repositories, and that the **Default download policy** is set to **Immediate** for custom repositories
   c. see Section 5.8, “Download Policies Overview” for more information.
2. Enable the content that you want to use (refer to Section 5.5, “Enabling Red Hat Repositories” .)
3. Synchronize the enabled content.
4. For the first export, perform a **complete Library** export so that all the synchronized content are exported. This generates content archives that you can later import into one or more disconnected Satellite Servers. For more information on performing a complete Library export, see Section 8.9, “Exporting the Library Environment” .
5. Export all future updates in the connected Satellite Servers incrementally. This generates leaner content archives that contain changes only from the recent set of updates. For example, if you enable and synchronize a new repository, the next exported content archive contains content only from the newly enabled repository. For more information on performing an incremental Library export, see Incremental Export.

On the disconnected Satellite Server

1. Copy the exported content archive that you want from the connected Satellite Server.
2. Import content to an organization using the procedure outlined in Section 8.10, “Importing into the Library Environment”. You can then manage content using Content Views or Lifecycle Environments as you require on the disconnected Satellite Server.

8.3.2. Connected Satellite Server for managing Content View Versions

In this scenario, your connected Satellite Server is receiving content from an external source like the CDN. The rest of your infrastructure is completely isolated, including a disconnected Satellite Server. However, the connected Satellite Server is not only used as a content store, but also is used to manage

content. Updates coming from the CDN are curated into Content Views and Lifecycle Environments. Once the content has been promoted to a designated Lifecycle Environment, the content can be exported and imported into the disconnected Satellite Server.

On the connected Satellite Server

1. Ensure that repositories are using the immediate download policy, either by:
   a. For existing repos using **On Demand** change their download policy on the repository details page to **Immediate**.
   b. For new repositories, ensure that the **Default Red Hat Repository download policy**setting is set to **Immediate** before enabling RH repos, and **Default download policy** is set to **Immediate** for custom repositories see Section 5.8, "Download Policies Overview" for more information.
2. Enable any content that you want to use (refer to Section 5.5, "Enabling Red Hat Repositories" .)
3. Synchronize the enabled content.
4. When you have new content republish the Content Views that include this content (see Chapter 7, *Managing Content Views*.) This should create a new Content View Version with the appropriate content to export.
5. For the first export, perform a **complete version** export on the Content View Version that you want to export. For more information see, Section 8.5, "Exporting a Content View Version" . This generates content archives that you can import into one or more disconnected Satellite Servers.
6. Export all future updates in the connected Satellite Servers incrementally. This generates leaner content archives that contain changes only from the recent set of updates. For example, if your Content View has a new repository, this exported content archive contains only the latest changes. For more information on performing an incremental export on Content View Versions, see Incremental Export.

On the disconnected Satellite Server

1. Copy the exported content archive from the connected Satellite Server.
2. Import the content to the organization that you want. For more information, see Section 8.8, "Importing a Content View Version". This will create a content view version from the exported content archives and them import content appropriately.

8.4. KEEPING TRACK OF YOUR EXPORTS

If you are exporting content to several disconnected Satellite Servers then using a **--destination-server** option provides a way to organize or maintain a record of what versions got exported to a given destination. For more information, see Section 8.6, "Keeping track of your exports" .

This option is available for all content-export operations. You can use the **destination-server** to

- Query what was previously exported to a given destination.
- Generate incremental exports automatically to the given destination server.

8.5. 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:

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

Prerequisites

To export a Content View, ensure that 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/exports** directory has free storage space equivalent to the size of the repositories 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 5.8, “Download Policies Overview”.
- Ensure that you synchronize Products that you export to the required date.
- Ensure that the user exporting the content has the **Content Exporter** role.

To Export a Content View Version:

1. List versions of the Content View that are available for export:

```
# hammer content-view version list \
--organization=export-org \
 --content-view=view

---|----------|---------|-------------|----------------------
ID | NAME     | VERSION | DESCRIPTION | LIFECYCLE ENVIRONMENTS
---|----------|---------|-------------|----------------------
5  | view 3.0 | 3.0     |             | Library
4  | view 2.0 | 2.0     |             |
3  | view 1.0 | 1.0     |             |
---|----------|---------|-------------|----------------------
```

Export a Content View version

Get the version number of desired version. The following example targets version **1.0** for export.

```
# hammer content-export complete version \
--content-view=view --version=1.0 \
--organization=export-21527
```

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

```
# ls -lh /var/lib/pulp/exports/export-21527/view/1.0/2021-02-25T18-59-26-00-00/
```

2. You require all three files, for example, the **tar.gz** archive file, the **toc.json** and **metadata.json** to import the content successfully.

Export with chunking

In many cases, the exported archive content can be several gigabytes in size. You might want to split it into smaller sizes or chunks. You can use the **--chunk-size-gb** option with the **hammer content-export** command to handle this. The following example uses the **--chunk-size-gb=2** to split the archives into **2 GB** chunks.

```
# hammer content-export complete version --content-view=view --version=1.0 --organization=export-21527 --chunk-size-gb=2

# ls -lh  /var/lib/pulp/exports/export-21527/view/1.0/2021-02-25T21-15-22-00-00/
```

8.6. KEEPING TRACK OF YOUR EXPORTS

When importing content to several Satellite Servers, the --destination-server option is especially useful for keeping track of which content was exported and to where.

You can use this flag to let the exporting Satellite Server keep track of content in specific servers. The **--destination-server** option functions to indicate the destination server that your content is imported to. The following example uses **--destination-server=mirror1** to export content to **mirror1**. The archive is created on the exporting Satellite Server. However, a record of each export is also maintained. This can be very useful when incrementally exporting.

```
# hammer content-export complete version \
--content-view=view --version=1.0 \
--organization=export-21527 \
--destination-server=mirror1
```

Incremental Export

Exporting complete versions can be a very expensive operation on storage space and resources. Content View versions that have multiple Red Hat Enterprise Linux trees can occupy several gigabytes of the space on Satellite Server.

You can use the **Incremental Export** functionality to help reduce demands on your infrastructure. **Incremental Export** exports only content that changes from the previously exported version. Generally, incremental changes are smaller than full exports. In the following example, since version **1.0** has already been exported and the command targets version 2.0 for export. To use incremental export, complete the following steps.

```
# hammer content-export incremental version \
 --content-view=view \
```

```
 --version=2.0 \
 --organization=export-21527

# ls -lh /var/lib/pulp/exports/export-21527/view/2.0/2021-02-25T21-45-34-00-00/
```

8.7. EXAMINING THE EXPORTS

You can query on the exports that you previously have created via the **hammer content-export list** command.

```
hammer content-export list --organization=export-21527

---|--------------------|--------------------------------------------------------------------------------|-------------|-------------
---------|------------------------|------------------------|----------------------
ID | DESTINATION SERVER | PATH                                                                           | TYPE        |
CONTENT VIEW VERSION | CONTENT VIEW VERSION ID | CREATED AT             | UPDATED
AT
---|--------------------|--------------------------------------------------------------------------------|-------------|-------------
---------|------------------------|------------------------|----------------------
1  |                    | /var/lib/pulp/exports/export-21527/view/1.0/2021-02-25T18-59-26-00-00 | complete
| view 1.0             | 3                      | 2021-02-25 18:59:30 UTC | 2021-02-25 18:59:30 UTC
2  |                    | /var/lib/pulp/exports/export-21527/view/1.0/2021-02-25T21-15-22-00-00 | complete
| view 1.0             | 3                      | 2021-02-25 21:15:26 UTC | 2021-02-25 21:15:26 UTC
3  |                    | /var/lib/pulp/exports/export-21527/view/2.0/2021-02-25T21-45-34-00-00 |
incremental | view 2.0       | 4                      | 2021-02-25 21:45:37 UTC | 2021-02-25 21:45:37
UTC
---|--------------------|--------------------------------------------------------------------------------|-------------|-------------
---------|------------------------|------------------------|----------------------
```

8.8. IMPORTING A CONTENT VIEW VERSION

You can use the archive that the **hammer content-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 8.5, “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. The Custom Repositories, Products and Content Views are automatically created if they do not exist in the importing organization.

Prerequisites

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

1. The exported archive must be in a directory under **/var/lib/pulp/imports**.
2. The directory must have **pulp:pulp** permissions so that Pulp can read and write the **.json** files in that directory.
3. If there are any Red Hat repositories in the export archive, the importing organization’s manifest must contain subscriptions for the products contained within the export.
4. The user importing the content view version must have the 'Content Importer' Role.

Procedure

1. Copy the archived file with the exported Content View version to the **/var/lib/pulp/imports** directory on Satellite Server where you want to import.
2. Set the user:group permission of the archive files to **pulp:pulp**.

```
# chown -R pulp:pulp /var/lib/pulp/imports/2021-02-25T21-15-22-00-00/
```

3. Verify that the permission change occurs:

```
# ls -lh  /var/lib/pulp/imports/2021-02-25T21-15-22-00-00/
```

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

```
# hammer content-import version --organization=import-20639 \
                               --path=/var/lib/pulp/imports/2021-02-25T21-15-22-00-00/
```

   Note that you must enter the full path **/var/lib/pulp/imports/<path>**. Relative paths do not work.

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

```
# hammer content-view version list --content-view=view \
                                   --organization=import-20639
---|----------|---------|-------------|----------------------
ID | NAME     | VERSION | DESCRIPTION | LIFECYCLE ENVIRONMENTS
---|----------|---------|-------------|----------------------
7  | view 1.0 | 1.0     |             | Library
---|----------|---------|-------------|----------------------
```

8.9. EXPORTING THE LIBRARY ENVIRONMENT

You can export contents of all Yum repositories in the Library environment of an organization to an archive file from Satellite Server and use this archive file to create the same repositories in another Satellite Server or in another Satellite Server organization. The exported archive file contains the following data:

- A JSON file containing Content View version metadata
- An archive file containing all the repositories from the Library environment of the organization.

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

- Docker content

Prerequisites

To export the contents of the Library lifecycle environment of the organization, ensure that 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/exports** directory has free storage space equivalent to the size of the repositories being exported for temporary files created during the export process.
- Ensure that you set download policy to **Immediate** for all repositories within the Library lifecycle environment you export. For more information, see Section 5.8, "Download Policies Overview".
- Ensure that you synchronize Products that you export to the required date.

To Export the Library Content of an Organization:

Use the organization name or ID to export.

```
# hammer content-export complete library --organization=export-21527
```

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

```
# ls -lh /var/lib/pulp/exports/export-21527/Export-Library/1.0/2021-03-02T03-35-24-00-00
total 68M
-rw-r--r--. 1 pulp pulp 68M Mar  2 03:35 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-20210302_0335.tar.gz
-rw-r--r--. 1 pulp pulp 333 Mar  2 03:35 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-20210302_0335-toc.json
-rw-r--r--. 1 root root 443 Mar  2 03:35 metadata.json
```

2. You require all three files, for example, the **tar.gz**, the **toc.json** and the **metadata.json** file to be able to import.
3. A new Content View **Export-Library** is created in the organization. This content view contains all the repositories belonging to this organization. A new version of this Content View is published and exported automatically.

Export with chunking

In many cases the exported archive content may be several gigabytes in size. If you want to split it into smaller sizes or chunks. You can use the **--chunk-size-gb** flag directly in the export command to handle this. In the following example, you can see how to specify **--chunk-size-gb=2** to split the archives in **2 GB** chunks.

```
# hammer content-export complete library --organization=export-21527 --chunk-size-gb=2
[.....
.....] [100%]
Generated /var/lib/pulp/exports/export-21527/Export-Library/2.0/2021-03-02T04-01-25-00-00/metadata.json

# ls -lh /var/lib/pulp/exports/export-21527/Export-Library/2.0/2021-03-02T04-01-25-00-00/
```

Incremental Export

Exporting Library content can be a very expensive operation in terms of space and resources. Organization that have multiple RHEL trees may occupy several gigabytes of the space on Satellite Server.

Satellite Server offers **Incremental Export** to help with this scenario. **Incremental Export** exports only things that changed from the previous export. These would be typically smaller than the full exports. In the example below we will incrementally export what changed from the previous export of all the

repositories in the Library lifecycle environment.

```
# hammer content-export incremental library --organization=export-21527
[.....
.....] [100%]
Generated /var/lib/pulp/exports/export-21527/Export-Library/3.0/2021-03-02T04-22-14-00-00/metadata.json
# ls -lh /var/lib/pulp/exports/export-21527/Export-Library/3.0/2021-03-02T04-22-14-00-00/
total 172K
-rw-r--r--. 1 pulp pulp 161K Mar  2 04:22 export-436882d8-de5a-48e9-a30a-17169318f908-20210302_0422.tar.gz
-rw-r--r--. 1 pulp pulp  333 Mar  2 04:22 export-436882d8-de5a-48e9-a30a-17169318f908-20210302_0422-toc.json
-rw-r--r--. 1 root root  492 Mar  2 04:22 metadata.json
```

1. Since nothing changed between the previous export and now in the Organization's library environment the change files are really small.

8.10. IMPORTING INTO THE LIBRARY ENVIRONMENT

You can use the archive that the **hammer content-export** command outputs to import into the Library lifecycle environment of another organization For more information about exporting contents from the Library environment, see Section 8.9, "Exporting the Library Environment".

Prerequisites

To import in to an Organization's library lifecycle environment ensure that Satellite Server where you want to import meets the following conditions:

1. The exported archive must be in a directory under **/var/lib/pulp/imports**.
2. The directory must have **pulp:pulp** permissions so that Pulp can read and write the **.json** files in that directory.
3. If there are any Red Hat repositories in the export archive, the importing organization's manifest must contain subscriptions for the products contained within the export.
4. The user importing the content must have the 'Content Importer' Role.

Procedure

1. Copy the archived file with the exported Content View version to the **/var/lib/pulp/imports** directory on Satellite Server where you want to import.
2. Set the permission of the archive files to **pulp:pulp**.

```
# chown -R pulp:pulp /var/lib/pulp/imports/2021-03-02T03-35-24-00-00
# ls -lh /var/lib/pulp/imports/2021-03-02T03-35-24-00-00
total 68M
-rw-r--r--. 1 pulp pulp 68M Mar  2 04:29 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-20210302_0335.tar.gz
-rw-r--r--. 1 pulp pulp 333 Mar  2 04:29 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-20210302_0335-toc.json
-rw-r--r--. 1 pulp pulp 443 Mar  2 04:29 metadata.json
```

3. On Satellite Server where you want to import, create/enable repositories the same name and label as the exported content.

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. Identify the Organization that you wish to import into.

6. To import the Library content to Satellite Server, enter the following command:

```
# hammer content-import library --organization=import-32158 \
                              --path=/var/lib/pulp/imports/2021-03-02T03-35-24-00-00
[.....................................................................................................................
.....................................................] [100%]
```

   Note you must enter the full path **/var/lib/pulp/imports/<path>**. Relative paths do not work.

7. To verify that you imported the Library content, check the contents of the Product and Repositories. A new Content View called **Import-Library** is created in the target organization. This Content View is used to facilitate the library content import.

8.11. EXPORTING A REPOSITORY

You can export the content of a repository in the Library environment of an organization from Satellite Server. You can use this archive file to create the same repository in another Satellite Server or in another Satellite Server organization.

You can export the following content from Satellite Server:

- RPM repositories
- Kickstart repositories
- Ansible repositories
- file repositories

You cannot export Docker content from Satellite Server.

The export contains the following data:

- Two JSON files containing repository metadata.
- One or more archive files containing the contents of the repository from the Library environment of the organization.

You need all the files, **tar.gz**, **toc.json** and **metadata.json**, to be able to import.

Prerequisites

To export the contents of a repository, ensure that Satellite Server from which you want to export, meets the following conditions:

- Ensure that the export directory has enough free storage space to accommodate the export.
- Ensure that the **/var/lib/pulp/exports** directory has enough free storage space equivalent to the size of the repositories being exported for temporary files created during the export process.

- Ensure that you set download policy to **Immediate** for the repository within the Library lifecycle environment you export. For more information, see Section 5.8, "Download Policies Overview".
- Ensure that you synchronize Products that you export to the required date.

Export a Repository

1. Use the repository name or ID to export.

```
# hammer content-export complete repository \
 --organization="My_Organization" \
 --product="My_Product" \
 --name="My_Repository"
```

2. Optional: Verify that the exported archive is located in the export directory:

```
# ls -lh /var/lib/pulp/exports/My_Organization/Export-My_Repository/1.0/2021-03-02T03-35-24-00-00
total 68M
-rw-r--r--. 1 pulp pulp 68M Mar  2 03:35 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-20210302_0335.tar.gz
-rw-r--r--. 1 pulp pulp 333 Mar  2 03:35 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-20210302_0335-toc.json
-rw-r--r--. 1 root root 443 Mar  2 03:35 metadata.json
```

Export a Repository with Chunking

In many cases the exported content archive may be several gigabytes in size. If you want to split it into chunks of smaller size, you can use the **--chunk-size-gb** argument in the export command and limit the size by an integer value in gigabytes.

1. Export content into archive chunks of a limited size, such as 2 GB:

```
# hammer content-export complete repository \
--chunk-size-gb=2 \
--organization="My_Organization" \
--product="My_Product" \
--name="My_Repository"

Generated /var/lib/pulp/exports/My_Organization/Export-My_Repository/1.0/2021-03-02T03-35-24-00-00/metadata.json
```

2. Optional: View the exported data:

```
# ls -lh /var/lib/pulp/exports/My_Organization/Export-My_Repository/1.0/2021-03-02T03-35-24-00-00/
```

8.12. EXPORTING A REPOSITORY INCREMENTALLY

Exporting a repository can be a very expensive operation in terms of system resources. A typical Red Hat Enterprise Linux tree may occupy several gigabytes of space on Satellite Server.

In such cases, you can use **Incremental Export** to export only pieces of content that changed since the previous export. Incremental exports typically result in smaller archive files than the full exports.

The example below shows incremental export of a repository in the Library lifecycle environment.

Procedure

1. Create an incremental export:

```
# hammer content-export incremental repository \
 --organization="My_Organization" \
 --product="My_Product" \
 --name="My_Repository"

Generated /var/lib/pulp/exports/My_Organization/Export-My_Repository/3.0/2021-03-02T03-35-24-00-00/metadata.json
```

2. Optional: View the exported data:

```
# ls -lh /var/lib/pulp/exports/My_Organization/Export-My_Repository/3.0/2021-03-02T03-35-24-00-00/
total 172K
-rw-r--r--. 1 pulp pulp  20M Mar  2 04:22 export-436882d8-de5a-48e9-a30a-17169318f908-20210302_0422.tar.gz
-rw-r--r--. 1 pulp pulp  333 Mar  2 04:22 export-436882d8-de5a-48e9-a30a-17169318f908-20210302_0422-toc.json
-rw-r--r--. 1 root root  492 Mar  2 04:22 metadata.json
```

8.13. IMPORTING A REPOSITORY

You can use the archive that the **hammer content-export** command outputs to import a repository in another organization. For more information about exporting content of a repository, see Section 8.11, “Exporting a Repository”.

Prerequisites

To import a repository to an Organization’s library lifecycle environment ensure that Satellite Server on which you want to import, meets the following conditions:

- The export files must be in a directory under **/var/lib/pulp/imports**.
- The directory must have **pulp:pulp** ownership so that Pulp can read and write the **.json** files in that directory.
- If the export contains any Red Hat repositories, the manifest of the importing organization must contain subscriptions for the products contained within the export.
- The user importing the content must have the *Content Importer* Role.

Procedure

1. Copy the export files to the **/var/lib/pulp/imports** directory on Satellite Server where you want to import.
2. Set the ownership of the export files to **pulp:pulp**.

```
# chown -R pulp:pulp /var/lib/pulp/imports/2021-03-02T03-35-24-00-00
# ls -lh /var/lib/pulp/imports/2021-03-02T03-35-24-00-00
```

```
total 68M
-rw-r--r--. 1 pulp pulp 68M Mar  2 04:29 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-20210302_0335.tar.gz
-rw-r--r--. 1 pulp pulp 333 Mar  2 04:29 export-1e25417c-6d09-49d4-b9a5-23df4db3d52a-20210302_0335-toc.json
-rw-r--r--. 1 pulp pulp 443 Mar  2 04:29 metadata.json
```

3. Identify the Organization that you wish to import into.

4. To import the Library content to Satellite Server, enter the following command:

```
# hammer content-import repository \
--organization="My_Organization" \
--path=/var/lib/pulp/imports/2021-03-02T03-35-24-00-00
```

Note that you must enter the full path **/var/lib/pulp/imports/*My_Exported_Repo_Dir***. Relative paths do not work.

5. To verify that you imported the repository, check the contents of the Product and Repository.

8.14. IMPORT/EXPORT CHEAT SHEET

Table 8.1. Export

Intent	Command
Fully Export a Content View version	**hammer content-export complete version --content-view="*My_Content_View*" --version=1.0 --organization="*My_Organization*"**
Incrementally Export a Content View version (assuming you have already exported something)	**hammer content-export incremental version --content-view="*My_Content_View*" --version=2.0 --organization="*My_Organization*"**
Fully Export a Repository	**hammer content-export complete repository --product="*My_Product*" --name="*My_Repository*" --organization="*My_Organization*"**
Incrementally Export a Repository (assuming you have already exported something)	**hammer content-export incremental repository --product="*My_Product*" --name="*My_Repository*" --organization="*My_Organization*"**
Fully Export an Organization's Library	**hammer content-export complete library --organization="*My_Organization*"**
Incrementally Export an Organization's Library (assuming you have already exported something)	**hammer content-export incremental library --organization="*My_Organization*"**

Intent	Command
Export a Content View version promoted to the Dev Environment	**hammer content-export complete version --content-view="*My_Content_View*" --organization="*My_Organization*" --lifecycle-environment="Dev"**
Export a Content View in smaller chunks (2-GB slabs)	**hammer content-export complete version --content-view="*My_Content_View*" --version=1.0 --organization="*My_Organization*" --chunk-size-gb=2**
Get a list of exports	**hammer content-export list --content-view="*My_Content_View*" --organization="*My_Organization*"**

Table 8.2. Import

Intent	Command
Import to a Content View version	**hammer content-import version --organization="*My_Organization*" --path="/var/lib/pulp/imports/*My_Exported_Repo_Dir*"**
Import a Repository	**hammer content-import repository --organization="*My_Organization*" --path="/var/lib/pulp/imports/*My_Exported_Repo_Dir*"**
Import to an Organization's Library	**hammer content-import library --organization="*My_Organization*" --path="/var/lib/pulp/imports/*My_Exported_Repo_Dir*"**

CHAPTER 9. MANAGING ACTIVATION KEYS

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

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

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
- System purpose

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.

Using the Same Activation Key with Multiple Content Hosts

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.

Using Multiple Activation Keys with a Content Host

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. You can specify the order of precedence by setting a host group parameter as follows:

```
$ hammer hostgroup set-parameter \
--name kt_activation_keys \
--value name_of_first_key, name_of_second_key,... \
--hostgroup hostgroup_name
```

9.1. 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?
- For Red Hat Enterprise Linux 8 hosts: Is there system purpose set on the activation key?

Note that Satellite automatically attaches subscriptions only for the products installed on a host. For subscriptions that do not list products installed on Red Hat Enterprise Linux by default, such as the Extended Update Support (EUS) subscription, use an activation key specifying the required subscriptions and with the auto-attach disabled.

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

1. Activation key that attaches subscriptions automatically.
   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. For Red Hat Enterprise Linux 8 hosts, you can configure the activation key to set system purpose on hosts during registration to enhance the automatic subscriptions attachment.

2. Activation key providing a custom set of subscription 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. Setting system purpose on the activation key does not affect this scenario.

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. Setting system purpose on the activation key does not affect this scenario.

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. To use the CLI instead of the web UI, see the CLI procedure.

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 intend to use the deprecated **Katello Agent** instead of **Remote Execution**, the Content View must contain the Satellite Tools 6.10 repository because it contains the **katello-agent** package.

7. Click **Save**.

8. Optional: For Red Hat Enterprise Linux 8 hosts, in the **System Purpose** section, you can configure the activation key with system purpose to set on hosts during registration to enhance subscriptions auto attachment.

CLI procedure

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. Optional: For Red Hat Enterprise Linux 8 hosts, enter the following command to configure the activation key with system purpose to set on hosts during registration to enhance subscriptions auto attachment.

```
# hammer activation-key update \
--organization "My_Organization" \
--name "My_Activation_Key" \
--service-level "Standard" \
--purpose-usage "Development/Test" \
--purpose-role "Red Hat Enterprise Linux Server" \
--purpose-addons "addons"
```

3. Obtain a list of your subscription IDs:

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

4. 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"
```

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

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

6. Override the default auto-enable status for the Satellite Tools 6.10 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-6.10-rpms \
--value 1 \
--organization "My_Organization"
```

9.2. UPDATING SUBSCRIPTIONS ASSOCIATED WITH AN ACTIVATION KEY

Use this procedure to change the subscriptions associated with an activation key. To use the CLI instead of the web UI, see the CLI procedure.

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

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**.

CLI procedure

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.

9.3. 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"
```

Multiple Activation Keys

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

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

Settings Conflicts

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

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

9.4. 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 physical subscription, but to inherit a host-based subscription from the hypervisor. For more information, see *Configuring Virtual Machine Subscriptions in Red Hat Satellite* .

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

Procedure

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**.

CLI procedure

- Enter the following command to enable auto-attach on the activation key:

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

9.5. 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

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**.

CLI procedure

- Enter the following command to set a default service level to Premium on the activation key:

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

CHAPTER 10. CONVERTING A HOST TO RED HAT ENTERPRISE LINUX WITH CONVERT2RHEL

Convert2RHEL enables the conversion of Red Hat Enterprise Linux derivative distributions into a supportable Red Hat Enterprise Linux state while retaining existing applications and configurations.

The conversion process is similar to a minor release upgrade of Red Hat Enterprise Linux in which every RPM package on the system is replaced. Third-party packages and non-Red Hat packages not available in Red Hat Enterprise Linux are not replaced.

Convert2RHEL removes unnecessary packages such as logos, or packages known to cause issues during the conversion, and replaces the **CentOS-release** package with the **rhel-release** package. It subscribes the system to Red Hat Satellite and replaces all packages signed by CentOS. For Satellite, it also subscribes the system to Red Hat Subscription Management.

The amount of time taken to do this can vary depending on how many packages need to be replaced, network speed, storage speed, and similar variables.

Use the global registration template to register and subscribe your system before the conversion. For more information, see Registering a Host to Red Hat Satellite Using the Global Registration Template in the *Managing Hosts* guide.

To convert hosts to Red Hat Enterprise Linux with convert2RHEL, you must complete the following procedures:

1. Section 10.1, “Installing Convert2RHEL”.
2. Section 10.2, “Creating an Activation Key for the Hosts”.
3. Section 10.3, “Preparing the Host Group”.
4. Section 10.4, “Registering Hosts for Conversion”.
5. Section 10.5, “Converting Hosts with Convert2RHEL”

Prerequisites

- You must have completed the steps listed in Preparing for a RHEL conversion.
- If you are providing the organization ID and activation key to register and subscribe your system, you must have created an activation key in Red Hat Satellite. For Satellite you can also create an activation key in Red Hat Subscription Manager. For more information, see Managing Activation Keys in this guide, and Understanding activation keys in Red Hat Subscription Manager documentation.
- Ensure you have a subscription manifest uploaded to your Satellite and that there are sufficient Red Hat Enterprise Linux subscriptions for the conversion.
- If you are providing the organization ID and activation key to register and subscribe your system, you must have created an activation key in Red Hat Satellite or Red Hat Subscription Manager. For more information, see Managing Activation Keys in this guide, and Understanding activation keys in Red Hat Subscription Manager documentation.
- Ensure you have a subscription manifest uploaded to your Satellite and that there are sufficient Red Hat Enterprise Linux subscriptions for the conversion.

- itIf you are accessing Red Hat Enterprise Linux packages using Satellite Server, you must use the organization ID and activation key to run the **convert2rhel** command.

10.1. INSTALLING CONVERT2RHEL

The following procedure makes Convert2RHEL available to the hosts you are converting.

Procedure

1. In the Satellite web UI, navigate to **Content** > **Products** and click **Create Product**.
2. In the **Name** field, enter the name of the product. Satellite automatically completes the **Label** field based on your entry 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. You do not need to complete **SSL Client Cert** or **SSL Client Key**.
6. Optional: From the **Sync Plan** list, select an existing sync plan or click **Create Sync Plan** to create a sync plan for your product requirements.
7. In the **Description** field, enter a description of the product.
8. Click **Save**.
9. Click the **Repositories** tab.
10. To add a repository, click **New Repository**.
11. In the **Name** field, enter the name of the repository.
12. From the **Type** list, set the content type to Yum.
13. Do not restrict the architecture or the OS version.
14. Use https://cdn.redhat.com/content/public/convert2rhel/7/x86_64/os/ as the upstream URL.
15. Disable the **Verify SSL** checkbox.
16. Optional: From the **GPG Key** list, select the GPG key for the product.
17. Optional: From the **SSL CA Cert** list, select the SSL CA certificate for the product.
    You do not need to complete **SSL Client Cert** or **SSL Client Key**.
18. Click **Save**.
19. Click the checkbox next to the new repository, then click **Sync Now**.

10.2. CREATING AN ACTIVATION KEY FOR THE HOSTS

Use this procedure to create an activation key for the hosts to use.

Procedure

1. In the Satellite web UI, navigate to **Content > Red Hat Repositories**.
2. Enable the Red Hat Enterprise Linux 7 Server repository.
3. In the Satellite web UI, navigate to **Content > Activation keys** and click **Create Activation Key**.
4. In the **Name** field, enter the name of the host activation key.
5. 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.
6. In the **Description** field, enter a description for the activation key.
7. From the **Environment** list, select the environment to use.
8. From the **Content View** list, select a Content View to use.
9. Click **Save**.
10. If SCA is not enabled:
    a. Click the **Subscriptions** tab, then click the **Add** submenu.
    b. Click the checkbox under the subscription you created before.
    c. Click **Add Selected**.
11. If SCA is enabled:
    a. In the **Repository Sets** tab, ensure only your named repository is enabled.

10.3. PREPARING THE HOST GROUP

Use this procedure to prepare a host group to convert CentOS 7 machines.

Procedure

1. In the Satellite web UI, navigate to **Configure > Host groups** and click **Create Host Group**.
2. In the **Name** field, enter a name for the host group.
3. In the **Description** field, enter a description of the host group.
4. In the **Lifecycle Environment** field, select **Library**.
5. Optional: In the **Content view** field enter a value for the Content view.
6. For Steps 7 - 10, Red Hat does not recommend setting values for this scenario as they might modify the host in ways that are not intended.
7. Optional: In the **Puppet Environment** field, select value the Puppet Environment.
8. The **Deploy on** field can be safely ignored.
9. Optional: In the **Puppet Capsule** field, select the name of the Puppet server configured for this capsule.

10. Optional: In the **Puppet CA Capsule** field, select the Puppet Server CA configured for this capsule.
11. Switch to the **Activation Keys** tab.
12. Optional: In the **OpenSCAP Capsule** field, select the OpenSCAP capsule to use for fetching SCAP content and uploading ARF reports.
13. Click **Submit**.
14. Switch to the **Activation Keys** tab.
15. In the **Activation Keys** field, enter the name of the product activation key.
16. Click **Submit**.

10.4. REGISTERING HOSTS FOR CONVERSION

Use this procedure to register CentOS hosts to Satellite.

Procedure

1. In the Satellite web UI, navigate to **Hosts > All Hosts** and click **Register Host**.
2. In the **Host Group** field, select the Host Group you want to use as a template for the conversion.
3. Optional, In the **Operating System** field, select the Operating System you are converting from.
4. Optional: In the **Capsule** field, select the required capsule.
5. Optional: Use the **Setup Insights** field to select whether to install Red Hat Insights on converted hosts.
6. Optional: Use the **Remote execution** field to select whether to install SSH on converted hosts.
7. Optional: In the **Token lifetime (hours)** field, set the lifetime. The default value is four hours. If you require more time to run the command on your hosts, increase the value of the Token Lifetime.
8. Optional: Check the **Insecure** box if the hosts might not trust the Satellite CA.
9. Optional: In the **Remote Execution Interface** field, if the host has multiple interfaces and remote execution must use a single one, select the appropriate network interface.
10. In the **Activation Key(s)** field, enter the name of the content activation key.
11. Click **Generate Command**.

When you run the command on the hosts you want to convert, it is registered in **Hosts > All Hosts**. If the host subscription status warns of Unknown subscription status, it can be ignored.

10.5. CONVERTING HOSTS WITH CONVERT2RHEL

Use the following procedure to perform the conversion.

Procedure

1. In the Satellite web UI, navigate to **Hosts > All Hosts** and select the target hosts you want to convert. You can use the **Search** field to filter the host list.
2. From the **Select Action** list, select **Schedule Remote Job**.
3. In the **Job category** field, select **Convert to RHEL**.
4. In the **Job template** field, verify that the Job Template is set to **Convert to RHEL**.
5. On the **Job Invocation** page, define the main job settings.
   The **Search Query** field displays the name of any search you have performed, or the list of hosts you have selected.
6. In the **Activation Key** field, enter the host activation key.
7. Optional: Click **Show Advanced Fields** if you want to configure further aspects.
8. Click **Submit**.
9. Note that:
   - The conversion can take some time.
   - During the conversion, the host reboots.
   - After the reboot, new OS facts are sent to Satellite.
   - When complete, the OS for the host changes to Red Hat Enterprise Linux and the subscription status turns green.

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. To use the CLI instead of the web UI, see the CLI procedure.

Procedure

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 applicable errata in 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.

CLI procedure

- To view errata that are available for all organizations, enter the following command:

  ```
  # hammer erratum list
  ```

- To view details of a specific erratum, enter the following command:

  ```
  # hammer erratum info --id erratum_ID
  ```

- You can search errata by entering the query with the **--search** option. For example, to view applicable errata for the selected product that contains the specified bugs ordered so that the security errata are displayed on top, enter the following command:

  ```
  # hammer erratum list \
  --product-id 7 \
  --search "bug = 1213000 or bug = 1207972" \
  --errata-restrict-applicable 1 \
  --order "type desc"
  ```

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*
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 Configuring Email Notifications in the *Administering Red Hat Satellite* guide.

11.3. LIMITATIONS TO REPOSITORY DEPENDENCY RESOLUTION

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.

To use the CLI instead of the web UI, see the CLI procedure.

Prerequisites

- A Content View with the repositories that contain required errata is created. For more information, see Section 7.1, "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.

CLI procedure

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.

To use the CLI instead of the web UI, see the CLI procedure.

Procedure

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.

CLI procedure

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 5.6, "Syncing Repositories".
- Register the host to an environment and Content View on Satellite Server. For more information, see Registering Hosts in the *Managing Hosts* guide.
- Configure the host for remote execution. For more information about running remote execution jobs, see Configuring and Setting up Remote Jobs in the *Managing Hosts* guide.

NOTE

If the host is already configured to receive content updates with the deprecated Katello Agent, migrate to remote execution instead. For more information, see Migrating from Katello Agent to Remote Execution 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.

CLI procedure

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. Using **Remote Execution**

```
# hammer job-invocation create \
--feature katello_errata_install \
--inputs errata=ERRATUM_ID1,ERRATUM_ID2 \
--search-query "name = client.example.com"
```

Using **Katello Agent** (deprecated)

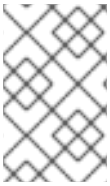
```
# hammer host errata apply --host "client.example.com" \
--errata-ids ERRATUM_ID1,ERRATUM_ID2...
```

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 5.6, “Syncing Repositories”.
- Register the hosts to an environment and Content View on Satellite Server. For more information, see Registering Hosts in the *Managing Hosts* guide.
- Configure the host for remote execution. For more information about running remote execution jobs, see Configuring and Setting up Remote Jobs in the *Managing Hosts* guide.



NOTE

If the host is already configured to receive content updates with the deprecated Katello Agent, migrate to remote execution instead. For more information, see Migrating from Katello Agent to Remote Execution 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**.

CLI procedure

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. Apply one of the errata to multiple hosts:
   Using **Remote Execution**

```
# hammer job-invocation create \
--feature katello_errata_install \
--inputs errata=ERRATUM_ID \
--search-query "applicable_errata = ERRATUM_ID"
```

   Using **Katello Agent** (deprecated)

   Identify 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"
```

   The following Bash script applies an erratum to each host for which this erratum is available:

```
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.

3. To see if an erratum is applied successfully, find the corresponding task in the output of the following command:

```
# hammer task list
```

4. View the state of a selected task:

```
# hammer task progress --id task_ID
```

11.8. APPLYING ERRATA TO A HOST COLLECTION

Using **Remote Execution**

```
# hammer job-invocation create \
--feature katello_errata_install \
--inputs errata=ERRATUM_ID1,ERRATUM_ID2,... \
--search-query "host_collection = HOST_COLLECTION_NAME"
```

Using **Katello Agent** (deprecated)

```
# hammer host-collection erratum install \
--errata "erratum_ID1,erratum_ID2,..." \
--name "host_collection_name"\
--organization "Your_Organization"
```

CHAPTER 12. MANAGING CONTAINER IMAGES

With Red Hat Satellite 6, you can import container images from various sources and distribute them to external containers using Content Views.

For information about containers, see Getting Started with Containers in *Red Hat Enterprise Linux Atomic Host 7*.

12.1. IMPORTING CONTAINER IMAGES

You can import container image repositories from Red Hat Registry or from other image registries.

This procedure uses repository discovery to find container images and import them as repositories. For more information about creating a product and repository manually, see Chapter 5, *Importing Content*.

To use the CLI instead of the web UI, see the CLI procedure.

Procedure

1. In the Satellite web UI, navigate to **Content** > **Products** and click **Repo Discovery**.
2. From the **Repository Type** list, select **Container Images**.
3. In the **Registry to Discover** field, enter the URL of the registry to import images from.
4. In the **Registry Username** field, enter the name that corresponds with your user name for the container image registry.
5. In the **Registry Password** field, enter the password that corresponds with the user name that you enter.
6. In the **Registry Search Parameter** field, enter any search criteria that you want to use to filter your search, and then click **Discover**.
7. Optional: To further refine the **Discovered Repository** list, in the **Filter** field, enter any additional search criteria that you want to use.
8. From the **Discovered Repository** list, select any repositories that you want to import, and then click **Create Selected**.
9. Optional: If you want to create a product, from the **Product** list, select **New Product**.
10. In the **Name** field, enter a product name.
11. Optional: In the **Repository Name** and **Repository Label** columns, you can edit the repository names and labels.
12. Click **Run Repository Creation**.
13. When repository creation is complete, you can click each new repository to view more information.
14. Optional: To filter the content you import to a repository, click a repository, and then navigate to **Limit Sync Tags**. Click to edit, and add any tags that you want to limit the content that synchronizes to Satellite.

15. Navigate to **Content** > **Products** and select the name of your product.

16. Select the new repositories and then click **Sync Now** to start the synchronization process.

To view the progress of the synchronization navigate to **Content** > **Sync Status** and expand the repository tree.

When the synchronization completes, you can click **Container Image Manifests** to list the available manifests. From the list, you can also remove any manifests that you do not require.

CLI procedure

1. Create the custom **Red Hat Container Catalog** product:

```
# hammer product create \
--name "Red Hat Container Catalog" \
--sync-plan "Example Plan" \
--description "Red Hat Container Catalog content" \
--organization "My_Organization"
```

2. Create the repository for the container images:

```
# hammer repository create \
--name "RHEL7" \
--content-type "docker" \
--url "http://registry.access.redhat.com/" \
--docker-upstream-name "rhel7" \
--product "Red Hat Container Catalog" \
--organization "My_Organization"
```

3. Synchronize the repository:

```
# hammer repository synchronize \
--name "RHEL7" \
--product "Red Hat Container Catalog" \
--organization "My_Organization"
```

12.2. MANAGING CONTAINER NAME PATTERNS

When you use Satellite to create and manage your containers, as the container moves through Content View versions and different stages of the Satellite lifecycle environment, the container name changes at each stage. For example, if you synchronize a container image with the name **ssh** from an upstream repository, when you add it to a Satellite product and organization and then publish as part of a Content View, the container image can have the following name: **my_organization_production-custom_spin-my_product-custom_ssh**. This can create problems when you want to pull a container image because container registries can contain only one instance of a container name. To avoid problems with Satellite naming conventions, you can set a registry name pattern to override the default name to ensure that your container name is clear for future use.

Limitations

If you use a registry name pattern to manage container naming conventions, because registry naming patterns must generate globally unique names, you might experience naming conflict problems. For example:

- If you set the **repository.docker_upstream_name** registry name pattern, you cannot publish or promote Content Views with container content with identical repository names to the **Production** lifecycle.
- If you set the **lifecycle_environment.name** registry name pattern, this can prevent the creation of a second container repository with the identical name.

You must proceed with caution when defining registry naming patterns for your containers.

Procedure

To manage container naming with a registry name pattern, complete the following steps:

1. In the Satellite web UI, navigate to **Content** > **Lifecycle Environments**, and either create a lifecycle environment or select a lifecycle environment to edit.
2. In the **Container Image Registry** area, click the edit icon to the right of **Registry Name Pattern** area.
3. Use the list of variables and examples to determine which registry name pattern you require.
4. In the **Registry Name Pattern** field, enter the registry name pattern that you want to use. For example, to use the **repository.docker_upstream_name**:

```
<%= repository.docker_upstream_name %>
```

5. Click **Save**.

12.3. MANAGING CONTAINER REGISTRY AUTHENTICATION

By default, users must authenticate to access containers images in Satellite.

You can specify whether you want users to authenticate to access container images in Satellite in a lifecycle environment. For example, you might want to permit users to access container images from the **Production** lifecycle without any authentication requirement and restrict access the **Development** and **QA** environments to authenticated users.

Procedure

To manage the authentication settings for accessing containers images from Satellite, complete the following steps:

1. In the Satellite web UI, navigate to **Content** > **Lifecycle Environments** and select the lifecycle environment that you want to manage authentication for.
2. To permit unauthenticated access to the containers in this lifecycle environment, select the **Unauthenticated Pull** check box. To restrict unauthenticated access, clear the **Unauthenticated Pull** check box.
3. Click **Save**.

12.4. USING KATELLO CONTAINER REGISTRIES

Podman and Docker can be used to fetch content from Katello container registries.

Container Registries on Smart Proxies

On Capsules with content, the Container Gateway Capsule plugin acts as the container registry. It caches authentication information from Katello and proxies incoming requests to Pulp. The Container Gateway comes default on Capsules with content.

Procedure

Logging in to the container registry:

```
podman login satellite.example.com
```

Listing container images:

```
podman search satellite.example.com/
```

Pulling container images:

```
podman pull satellite.example.com/my-image:<optional_tag>
```

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.

To use the CLI instead of the web UI, see the CLI procedure.

Procedure

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**.

CLI procedure

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 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.

To use the CLI instead of the web UI, see the CLI procedure.

Procedure

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**.

CLI procedure

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 \
--id repo_ID \
--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.

To use the CLI instead of the web UI, see the CLI procedure.

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. Satellite 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. Satellite 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.
9. Optional: In the **Upstream Authentication Token** field, provide the token of the upstream repository user for authentication. Leave this field empty if the repository does not require authentication.
10. Optional: Check the **Mirror on Sync** check box to have this repository mirror the source repository during synchronization. It defaults to **true** (checked).
11. Optional: In the **HTTP Proxy Policy** field, select the desired HHTP proxy. The default value is **Global Default**.
12. Optional: Check the **Publish via HTTP** to have this repository published using HTTP during synchronization. It defaults to **true** (checked).
13. Optional: In the **GPG Key** field, select the GPG key for the repository.
14. Optional: In the **SSL CA Cert** field, select the SSL CA Certificate for the repository.
15. Optional: In the **SSL Client cert** field, select the SSL Client Certificate for the repository.
16. Optional: In the **SSL Client Key** field, select the SSL Client Key for the repository.
17. Click **Save**.

CLI procedure

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' or 'on_demand').
--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 6.10 repositories are enabled.

```
# subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-7-server-satellite-tools-6.10-rpms
```

2. Install the Pulp Manifest package:

```
# satellite-maintain packages install python3-pulp_manifest
```

Note that this command stops the Satellite service and re-runs satellite-installer. Alternatively, to prevent downtime caused by stopping the service, you can use the following:

```
# subscription-manager repos --enable rhel-7-server-satellite-capsule-6.10-rpms
# satellite-maintain packages unlock
# yum install install python-pulp-manifest -y
# satellite-maintain packages lock
# subscription-manager repos --disable rhel-7-server-satellite-capsule-6.10-rpms
```

This installs the package without downtime.

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. Satellite 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. Clear this field if the repository does not require authentication.
10. Optional: In the **Upstream Password** field, enter the corresponding password for your upstream user name.
11. Optional: In the **Upstream Authentication Token** field, provide the token of the upstream repository user for authentication. Leave this field empty if the repository does not require authentication.
12. Optional: Check the **Mirror on Sync** check box to have this repository mirror the source repository during synchronization. It defaults to **true** (checked).
13. Optional: In the **HTTP Proxy Policy** field, select the desired HHTP proxy. The default value is **Global Default**.
14. Optional: Check the **Publish via HTTP** to have this repository published using HTTP during synchronization. It defaults to **true** (checked).
15. Optional: In the **GPG Key** field, select the GPG key for the repository.
16. Optional: In the **SSL CA Cert** field, select the SSL CA Certificate for the repository.
17. Optional: In the **SSL Client cert** field, select the SSL Client Certificate for the repository.
18. Optional: In the **SSL Client Key** field, select the SSL Client Key for the repository.
19. 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 6.10 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 6.10 repositories are enabled.

```
# subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-7-server-satellite-tools-6.10-rpms
```

2. Install the Pulp Manifest package:

```
# yum install python3-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.
11. Optional: In the **Upstream Authentication Token** field, provide the token of the upstream repository user for authentication. Leave this field empty if the repository does not require authentication.
12. Optional: Check the **Mirror on Sync** check box to have this repository mirror the source repository during synchronization. It defaults to **true** (checked).
13. Optional: In the **HTTP Proxy Policy** field, select the desired HHTP proxy. The default value is **Global Default**.
14. Optional: Check the **Publish via HTTP** to have this repository published using HTTP during synchronization. It defaults to **true** (checked).
15. Optional: In the **GPG Key** field, select the GPG key for the repository.
16. Optional: In the **SSL CA Cert** field, select the SSL CA Certificate for the repository.
17. Optional: In the **SSL Client cert** field, select the SSL Client Certificate for the repository.
18. Optional: In the **SSL Client Key** field, select the SSL Client Key for the repository.
19. Click **Save**.
20. 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.
21. In the product's window, select the name of the repository you want to update.
22. 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

Use this procedure to upload files to a custom file type repository.

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. 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.

CLI procedure

```
# hammer repository upload-content \
--id repo_ID \
--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

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.

CLI procedure

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
```

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

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.

1. Create the NFS share. This example uses a share at **nfs.example.com:/satellite/pulp**. 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:

```
# satellite-maintain packages install nfs-utils
```

4. You need to copy the existing contents of **/var/lib/pulp** to the NFS share. First, mount the NFS share to a temporary location:

```
# mkdir /mnt/temp
# mount -o rw nfs.example.com:/satellite/pulp /mnt/temp
```

   Copy the existing contents of **/var/lib/pulp** to the temporary location:

```
# cp -r /var/lib/pulp/* /mnt/temp/.
```

5. Set the permissions for all files on the share to use the **pulp** user.
6. Unmount the temporary storage location:

```
# umount /mnt/temp
```

7. Remove the existing contents of **/var/lib/pulp**:

```
# rm -rf /var/lib/pulp/*
```

8. Edit the **/etc/fstab** file and add the following line:

```
nfs.example.com:/satellite/pulp    /var/lib/pulp   nfs
rw,hard,intr,context="system_u:object_r:pulpcore_var_lib_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**:

```
# df
Filesystem                     1K-blocks     Used Available Use% Mounted on
...
nfs.example.com:/satellite/pulp 309506048 58632800 235128224  20% /var/lib/pulp
...
```

Also confirm that the existing content exists at the mount on **var/lib/pulp**:

```
# ls /var/lib/pulp
```

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 to ensure the NFS share works as expected. For more information, see Section 5.6, “Syncing Repositories”.

APPENDIX B. CONFIGURING SATELLITE TO SYNCHRONIZE CONTENT WITH A LOCAL CDN SERVER

In a disconnected environment, you must ensure that Satellite Server contains the required content to provision systems with the latest security updates, errata, and packages. To do this, follow this procedure to download content ISO images from the Red Hat Customer Portal and import them into a local CDN server. You can host the local CDN server on the base operating system of Satellite Server or on a system that is accessible to Satellite over HTTP. Next, you must configure Satellite Server to synchronize content with the local CDN server.

Procedure

1. Log on 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 **Red Hat Enterprise Linux 7 Server (x86_64)**to download the ISO image.
5. Copy all of Satellite Content ISO images to a system that you want to use as a local CDN server. For example, the **/root/isos** directory on Satellite Server.
   Note that storing the content on the same system where Satellite is installed is not a requirement. The CDN can be hosted on a different system inside the same disconnected network as long as it is accessible to Satellite Server over HTTP.
6. On the system that you want to use as your local CDN server, create a local directory that is accessible over httpd. For example, **/var/www/html/pub/sat-import/**:

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

7. Create a mount point and temporarily mount the ISO image at that location:

```
# mkdir /mnt/iso
# mount -o loop /root/isos/first_iso /mnt/iso
```

8. Recursively copy content of the first ISO image to the local directory:

```
# cp -ruv /mnt/iso/* /var/www/html/pub/sat-import/
```

9. If you do not plan to use the mounted binary DVD ISO image, unmount and remove the mount point:

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

10. 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/**.
11. Ensure that the SELinux context for the directory is correct:

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

12. In the Satellite web UI, navigate to **Content** > **Subscriptions**.
13. Click **Manage Manifest**.
14. Edit the **Red Hat CDN URL**field to point to the host name of the system that you use as a local CDN server with the newly created directory, for example:
http://server.example.com/pub/sat-import/
15. Click **Update** and then upload your manifest into Satellite.

APPENDIX C. 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.

To import Kickstart repositories for Red Hat Enterprise Linux 7, complete Section C.1, "Importing Kickstart Repositories for Red Hat Enterprise Linux 7".

To import Kickstart repositories for Red Hat Enterprise Linux 8, complete Section C.2, "Importing Kickstart Repositories for Red Hat Enterprise Linux 8".

C.1. IMPORTING KICKSTART REPOSITORIES FOR RED HAT ENTERPRISE LINUX 7

To import Kickstart repositories for Red Hat Enterprise Linux 7, complete the following steps on Satellite.

Procedure

1. Navigate to the Red Hat Customer Portal at https://access.redhat.com/ and log in.
2. In the upper left of the window, click **Downloads**.
3. To the right of **Red Hat Enterprise Linux 7**, click **Versions 7 and below**.
4. From the **Version** list, select the required version of the Red Hat Enterprise Linux 7, for example 7.7.
5. In the Download Red Hat Enterprise Linux window, locate the binary DVD version of the ISO image, for example, **Red Hat Enterprise Linux 7.7 Binary DVD**, and click **Download Now**.
6. When the download completes, copy the ISO image to Satellite Server.
7. On Satellite Server, create a mount point and temporarily mount the ISO image at that location:

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

8. Create Kickstart directories:

```
# mkdir --parents \
/var/www/html/pub/sat-import/content/dist/rhel/server/7/7.7/x86_64/kickstart/
```

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

```
# cp -a /mnt/iso/* /var/www/html/pub/sat-
import/content/dist/rhel/server/7/7.7/x86_64/kickstart/
```

10. Add the following entries to the listing files:
    To the **/var/www/html/pub/sat-import/content/dist/rhel/server/7/listing** file, append the version number with a new line. For example, for the RHEL 7.7 ISO, append **7.7**.

To the **/var/www/html/pub/sat-import/content/dist/rhel/server/7/7.7/listing** file, append the architecture with a new line. For example, **x86_64**.

To the **/var/www/html/pub/sat-import/content/dist/rhel/server/7/7.7/x86_64/listing** file, append **kickstart** with a new line.

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

```
# cp /mnt/iso/.treeinfo \
/var/www/html/pub/sat-import/content/dist/rhel/server/7/7.7/x86_64/kickstart/treeinfo
```

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

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

13. In the Satellite web UI, enable the Kickstart repositories.

C.2. IMPORTING KICKSTART REPOSITORIES FOR RED HAT ENTERPRISE LINUX 8

To import Kickstart repositories for Red Hat Enterprise Linux 8, complete the following steps on Satellite.

Procedure

1. Navigate to the Red Hat Customer Portal at https://access.redhat.com/ and log in.
2. In the upper left of the window, click **Downloads**.
3. Click **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-binary-dvd.iso /mnt/iso
```

7. Create directories for Red Hat Enterprise Linux 8 AppStream and BaseOS Kickstart repositories:

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

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

8. 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.

9. Add the following entries to the listing files:
   To the **/var/www/html/pub/sat-import/content/dist/rhel8/8.1/x86_64/appstream/listing** file, append **kickstart** with a new line.

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

   To the **/var/www/html/pub/sat-import/content/dist/rhel8/listing** file, append the version number with a new line. For example, for the RHEL 8.1 binary ISO, append **8.1**.

10. 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
```

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

12. 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**

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

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

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

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

16. Save and close the file.

17. 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
```

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

19. 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**

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

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

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

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

23. Save and close the file.

24. 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
```

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

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

26. In the Satellite web UI, enable the Kickstart repositories.

APPENDIX D. REVERTING SATELLITE TO DOWNLOAD CONTENT FROM RED HAT CDN

If your environment changes from disconnected to connected, you can reconfigure a disconnected Satellite to download content directly from the Red Hat CDN.

Procedure

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 is now configured to download content from the CDN the next time that it synchronizes repositories.