Administration Guide: Uyuni {productnumber}

Administration Guide Overview

Updated: 2021-09-22

This book provides guidance on performing administration tasks on the Uyuni Server.

Image Building and Management

Image Building Overview

Uyuni enables system administrators to build containers and OS Images and push the result in image stores. The workflow looks like this:

1. Define an image store

2. Define an image profile and associate it with a source (either a git repository or a directory)

3. Build the image

4. Push the image to the image store

Uyuni supports two distinct build types: dockerfile, and the Kiwi image system.

The Kiwi build type is used to build system, virtual, and other images. The image store for the Kiwi build type is pre-defined as a file system directory at `/srv/www/os-images` on the server. Uyuni serves the image store over HTTPS from `//<SERVER-FQDN>/os-images/`. The image store location is unique and is not customizable.

Images are always stored in `/srv/www/os-image/<organization id>`.

Container Images
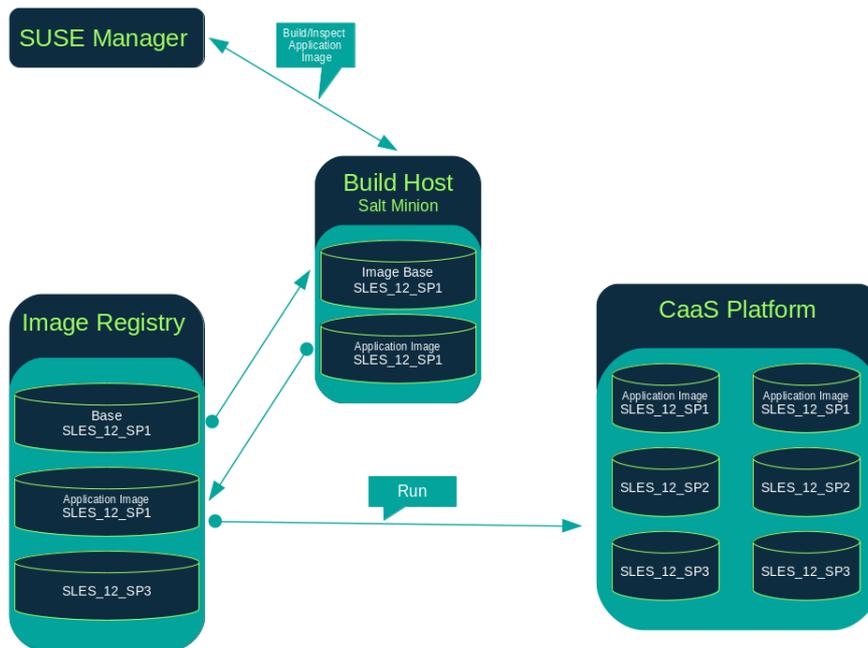
Requirements

The containers feature is available for Salt clients running {sles} 12 or later. Before you begin, ensure your environment meets these requirements:

- A published git repository containing a dockerfile and configuration scripts. The repository can be public or private, and should be hosted on GitHub, GitLab, or BitBucket.

- A properly configured image store, such as a Docker registry.

> **NOTE**
>
> If you require a private image registry you can use an open source solution such as `Portus`. For additional information on setting up Portus as a registry provider, see the Portus Documentation.

For more information on Containers or {caasp}, see:

- https://documentation.suse.com/sles/15-SP3/html/SLES-all/book-container.html

- https://documentation.suse.com/suse-caasp/4.5/

Create a Build Host

To build images with Uyuni, you need to create and configure a build host. Container build hosts are Salt clients running {sle} 12 or later. This section guides you through the initial configuration for a build host.

From the Uyuni {webui}, perform these steps to configure a build host:

1. Select a Salt client to be designated as a build host from the menu:Systems[Overview] page.

2. From the `System Details` page of the selected client assign the containers modules. Go to menu:Software[Software Channels] and enable the containers module (for example, `SLE-Module-Containers15-Pool` and `SLE-Module-Containers15-Updates`). Confirm by clicking btn:[Change Subscriptions].

3. From the menu:System Details[Properties] page, enable `Container Build Host` from the `Add-on System Types` list and confirm by clicking btn:[Update Properties].

4. Install all required packages by applying `Highstate`. From the system details page select menu:States[Highstate] and click `Apply Highstate`. Alternatively, apply Highstate from the Uyuni Server command line:

```
salt '$your_client' state.highstate
```

Create an Activation Key for Containers

The containers built using Uyuni use channels associated to the activation key as repositories when building the image. This section guides you through creating an ad-hoc activation key for this purpose.

NOTE To build a container, you need an activation key that is associated with a channel other than `SUSE Manager Default`.

1. Select menu:Systems[Activation Keys].

2. Click btn:[Create Key].

3. Enter a `Description` and a `Key` name. Use the drop-down menu to select the `Base Channel` to associate with this key.

4. Confirm with btn:[Create Activation Key].

For more information, see [bp.key.managment].

Create an Image Store

All built images are pushed to an image store. This section contains information about creating an image store.
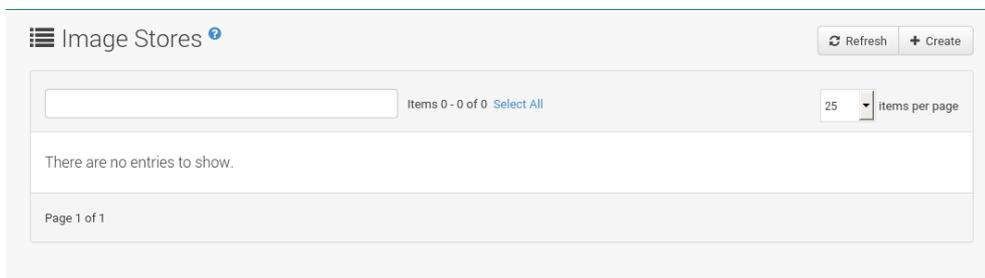
1. Select menu:Images[Stores].

2. Click `Create` to create a new store.

3. Define a name for the image store in the `Label` field.

4. Provide the path to your image registry by filling in the `URI` field, as a fully qualified domain name (FQDN) for the container registry host (whether internal or external).

```
registry.example.com
```

The Registry URI can also be used to specify an image store on a registry that is already in use.

```
registry.example.com:5000/myregistry/myproject
```

5. Click btn:[Create] to add the new image store.

Create an Image Profile

All container images are built using an image profile, which contains the building instructions. This section contains information about creating an image profile with the Uyuni {webui}.
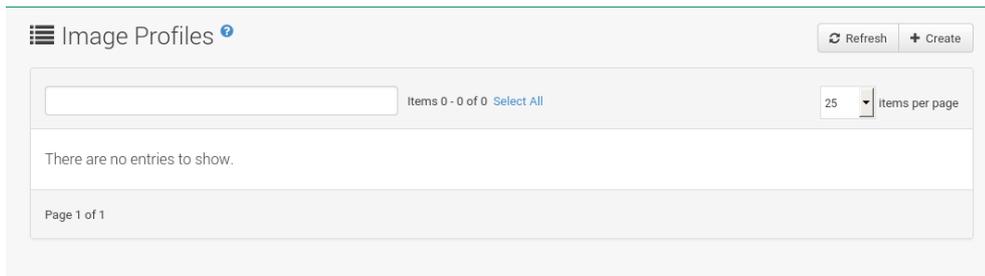
Procedure: Create an Image Profile

1. To create an image profile select menu:Images[Profiles] and click btn:[Create].

2. Provide a name for the image profile by filling in the `Label` field.

> **NOTE**
>
> If your container image tag is in a format such as `myproject/myimage`, make sure your image store registry URI contains the `/myproject` suffix.

3. Use a dockerfile as the `Image Type`.

4. Use the drop-down menu to select your registry from the `Target Image Store` field.

5. In the `Path` field, type a GitHub, GitLab or BitBucket repository URL. The URL should be be http, https, or a token authentication URL. Use one of these formats:

GitHub Path Options

- GitHub single user project repository

```
https://github.com/USER/project.git#branchname:folder
```

- GitHub organization project repository

```
https://github.com/ORG/project.git#branchname:folder
```

- GitHub token authentication

If your git repository is private, modify the profile's URL to include authentication. Use this URL format to authenticate with a GitHub token:

```
https://USER:<AUTHENTICATION_TOKEN>@github.com/USER/project.git#master:/container/
```

GitLab Path Options
- GitLab single user project repository

```
https://gitlab.example.com/USER/project.git#master:/container/
```

- GitLab groups project repository

```
https://gitlab.example.com/GROUP/project.git#master:/container/
```

- GitLab token authentication

If your git repository is private and not publicly accessible, you need to modify the profile's git URL to include authentication. Use this URL format to authenticate with a GitLab token:

```
https://gitlab-ci-
token:<AUTHENTICATION_TOKEN>@gitlab.example.com/USER/project.git#master:/container/
```

IMPORTANT   If you do not specify a git branch, the `master` branch is used by default. If a `folder` is not specified, the image sources (dockerfile sources) are expected to be in the root directory of the GitHub or GitLab checkout.

1. Select an `Activation Key`. Activation keys ensure that images using a profile are assigned to the correct channel and packages.

NOTE   When you associate an activation key with an image profile you are ensuring any image using the profile uses the correct software channel and any packages in the channel.

2. Click the btn:[Create] button.

Example Dockerfile Sources

An Image Profile that can be reused is published at https://github.com/SUSE/manager-build-profiles

NOTE	The `ARG` parameters ensure that the built image is associated with the desired repository served by Uyuni. The `ARG` parameters also allow you to build image versions of {sles} which may differ from the version of {sles} used by the build host itself. For example: The `ARG repo` parameter and the `echo` command pointing to the repository file, creates and then injects the correct path into the repository file for the desired channel version. The repository is determined by the activation key that you assigned to your image profile.

```
FROM registry.example.com/sles12sp2
MAINTAINER Tux Administrator "tux@example.com"

### Begin: These lines Required for use with {productname}

ARG repo
ARG cert

# Add the correct certificate
RUN echo "$cert" > /etc/pki/trust/anchors/RHN-ORG-TRUSTED-SSL-CERT.pem

# Update certificate trust store
RUN update-ca-certificates

# Add the repository path to the image
RUN echo "$repo" > /etc/zypp/repos.d/susemanager:dockerbuild.repo

### End: These lines required for use with {productname}

# Add the package script
ADD add_packages.sh /root/add_packages.sh

# Run the package script
RUN /root/add_packages.sh

# After building remove the repository path from image
RUN rm -f /etc/zypp/repos.d/susemanager:dockerbuild.repo
```

Using Custom Info Key-value Pairs as Docker Buildargs

You can assign custom info key-value pairs to attach information to the image profiles. Additionally, these key-value pairs are passed to the Docker build command as `buildargs`.

For more information about the available custom info keys and creating additional ones, see **[Reference › Systems ›].**

Build an Image

There are two ways to build an image. You can select menu:Images[Build] from the left navigation bar, or click the build icon in the menu:Images[Profiles] list.

Procedure: Building an Image

1. Select menu:Images[Build].

2. Add a different tag name if you want a version other than the default `latest` (only relevant to containers).

3. Select `Build Profile` and `Build Host`.

> **NOTE** Notice the `Profile Summary` to the right of the build fields. When you have selected a build profile, detailed information about the selected profile is displayed in this area.

4. To schedule a build click the btn:[Build] button.

Import an Image

You can import and inspect arbitrary images. Select menu:Images[Image List] from the left navigation bar. Complete the text boxes of the `Import` dialog. When it has processed, the imported image is listed on the `Image List` page.

Procedure: Importing an Image

1. From menu:Images[Image list] click btn:[Import] to open the `Import Image` dialog.

2. In the `Import Image` dialog complete these fields:

   Image store

   The registry from where the image is pulled for inspection.

   Image name

   The name of the image in the registry.

Image version

> The version of the image in the registry.

Build host

> The build host that pulls and inspects the image.

Activation key

> The activation key that provides the path to the software channel that the image is inspected with.

3. For confirmation, click btn:[Import].

The entry for the image is created in the database, and an `Inspect Image` action on Uyuni is scheduled.

When it has been processed, you can find the imported image in the `Image List`. It has a different icon in the `Build` column, to indicate that the image is imported. The status icon for the imported image can also be seen on the `Overview` tab for the image.

Troubleshooting

These are some known problems when working with images:

- HTTPS certificates to access the registry or the git repositories should be deployed to the client by a custom state file.

- SSH git access using Docker is currently unsupported.

OS Images

OS Images are built by the Kiwi image system. The output image is customizable and can be PXE, QCOW2, LiveCD, or other types of images.

For more information about the Kiwi build system, see the Kiwi documentation.

Requirements

The Kiwi image building feature is available for Salt clients running {sles} 12 and {sles} 11.

Kiwi image configuration files and configuration scripts must be accessible in one of these locations:

- Git repository

- HTTP hosted tarball

- Local build host directory

For an example of a complete Kiwi repository served by git, see https://github.com/SUSE/manager-build-profiles/tree/master/OSImage

NOTE You need at least 1 GB of RAM available for hosts running OS Images built with Kiwi. Disk space depends on the actual size of the image. For more information, see the documentation of the underlying system.

WARNING The build host must be a Salt client. Do not install the build host as a traditional client.

Create a Build Host

To build all kinds of images with Uyuni, create and configure a build host. OS Image build hosts are Salt clients running on {sles} 15 (SP2 or later), {sles} 12 (SP3 or later) or {sles} 11 SP4.

This procedure guides you through the initial configuration for a build host.

IMPORTANT The operating system on the build host must match the operating system on the targeted image.

For example, build {sles} 15 based images on a build host running {sles} 15 (SP2 or later) OS version. Build {sles} 12 based images on a build host running {sles} 12 SP4 or {sles} 12 SP3 OS version. Build {sles} 11 based images on a build host running {sles} 11 SP4 OS version.

Configure the build host in the Uyuni {webui}:

1. Select a client to be designated as a build host from the menu:Systems[Overview] page.

2. Navigate to the menu:System Details[Properties] tab, enable the `Add-on System Type OS Image Build Host`. Confirm with btn:[Update Properties].

3. Navigate to menu:System Details[Software > Software Channels], and enable the required software channels depending on the build host version.

   ° {sles} 11 build hosts require Uyuni Client tools (`SLE-Manager-Tools11-Pool` and `SLE-Manager-Tools11-Updates`).

   ° {sles} 12 build hosts require Uyuni Client tools (`SLE-Manager-Tools12-Pool` and `SLE-Manager-Tools12-Updates`).

   ° {sles} 15 build hosts require {sles} modules `SLE-Module-DevTools15-SP2-Pool` and `SLE-Module-DevTools15-SP2-Updates`. Schedule and click btn:[Confirm].

4. Install Kiwi and all required packages by applying `Highstate`. From the system details page select menu:States[Highstate] and click btn:[Apply Highstate]. Alternatively, apply Highstate from the Uyuni Server command line:

```
salt '$your_client' state.highstate
```

Uyuni Web Server Public Certificate RPM

Build host provisioning copies the Uyuni certificate RPM to the build host. This certificate is used for accessing repositories provided by Uyuni.

The certificate is packaged in RPM by the `mgr-package-rpm-certificate-osimage` package script. The package script is called automatically during a new Uyuni installation.

When you upgrade the `spacewalk-certs-tools` package, the upgrade scenario calls the package script using the default values. However if the certificate path was changed or unavailable, call the package

script manually using `--ca-cert-full-path` `<path_to_certificate>` after the upgrade procedure has finished.

Package script call example

```
/usr/sbin/mgr-package-rpm-certificate-osimage --ca-cert-full-path /root/ssl-build/RHN-ORG-
TRUSTED-SSL-CERT
```

The RPM package with the certificate is stored in a salt-accessible directory such as:

```
/usr/share/susemanager/salt/images/rhn-org-trusted-ssl-cert-osimage-1.0-1.noarch.rpm
```

The RPM package with the certificate is provided in the local build host repository:

```
/var/lib/Kiwi/repo
```

IMPORTANT

Specify the RPM package with the Uyuni SSL certificate in the build source, and make sure your Kiwi configuration contains `rhn-org-trusted-ssl-cert-osimage` as a required package in the `bootstrap` section.

config.xml

```
...
  <packages type="bootstrap">
    ...
    <package name="rhn-org-trusted-ssl-cert-osimage"
bootinclude="true"/>
  </packages>
...
```

Create an Activation Key for OS Images

Create an activation key associated with the channel that your OS Images can use as repositories when building the image.

Activation keys are mandatory for OS Image building.

NOTE

To build OS Images, you need an activation key that is associated with a channel other than `SUSE Manager Default`.

1. In the {webui}, select menu:Systems[Activation Keys].

2. Click `Create Key`.

3. Enter a `Description`, a `Key` name, and use the drop-down box to select a `Base Channel` to associate with the key.

4. Confirm with btn:[Create Activation Key].

For more information, see [bp.key.managment].

Create an Image Store

OS Images can require a significant amount of storage space. Therefore, we recommended that the OS Image store is located on a partition of its own or on a Btrfs subvolume, separate from the root partition. By default, the image store is located at `/srv/www/os-images`.

> **NOTE**
>
> Image stores for Kiwi build type, used to build system, virtual, and other images, are not supported yet.
>
> Images are always stored in `/srv/www/os-images/<organization id>` and are accessible via HTTP/HTTPS `https://<susemanager_host>/os-images/<organization id>`.

Create an Image Profile

Manage image profiles using the {webui}.
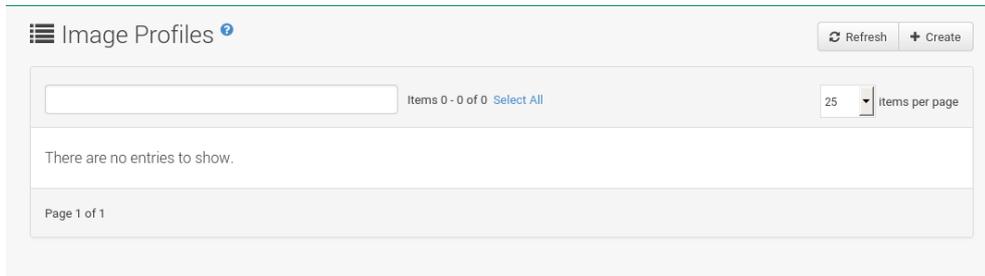
Procedure: Create an Image Profile

1. To create an image profile select from menu:Images[Profiles] and click btn:[Create].

2. In the `Label` field, provide a name for the `Image Profile`.

3. Use `Kiwi` as the `Image Type`.

4. Image store is automatically selected.

5. Enter a `Config URL` to the directory containing the Kiwi configuration files:

   a. git URI

   b. HTTPS tarball

   c. Path to build host local directory

6. Enter `Kiwi options` if needed. If the Kiwi configuration files specify multiple profiles, use `--profile <name>` to select the active one. For other options, see Kiwi documentation.

7. Select an `Activation Key`. Activation keys ensure that images using a profile are assigned to the correct channel and packages.

> **NOTE** Associate an activation key with an image profile to ensure the image profile uses the correct software channel, and any packages.

8. Confirm with the btn:[Create] button.

Source format options

- git/HTTP(S) URL to the repository

  URL to the git repository containing the sources of the image to be built. Depending on the layout of the repository the URL can be:

  ```
  https://github.com/SUSE/manager-build-profiles
  ```

  You can specify a branch after the `#` character in the URL. In this example, we use the `master` branch:

  ```
  https://github.com/SUSE/manager-build-profiles#master
  ```

  You can specify a directory that contains the image sources after the `:` character. In this example, we use `OSImage/POS_Image-JeOS6`:

  ```
  https://github.com/SUSE/manager-build-profiles#master:OSImage/POS_Image-JeOS6
  ```

- HTTP(S) URL to the tarball

  URL to the tar archive, compressed or uncompressed, hosted on the webserver.

  ```
  https://myimagesourceserver.example.org/MyKiwiImage.tar.gz
  ```

- Path to the directory on the build host

  Enter the path to the directory with the Kiwi build system sources. This directory must be present on the selected build host.

  ```
  /var/lib/Kiwi/MyKiwiImage
  ```

Example of Kiwi Sources

Kiwi sources consist at least of `config.xml`. Usually, `config.sh` and `images.sh` are present as well. Sources can also contain files to be installed in the final image under the `root` subdirectory.

For information about the Kiwi build system, see the Kiwi documentation.

{suse} provides examples of fully functional image sources at the SUSE/manager-build-profiles public GitHub repository.

Example of JeOS config.xml

```xml
<?xml version="1.0" encoding="utf-8"?>

<image schemaversion="6.1" name="POS_Image_JeOS6">
    <description type="system">
        <author>Admin User</author>
        <contact>noemail@example.com</contact>
        <specification>SUSE Linux Enterprise 12 SP3 JeOS</specification>
    </description>
    <preferences>
        <version>6.0.0</version>
        <packagemanager>zypper</packagemanager>
        <bootsplash-theme>SLE</bootsplash-theme>
        <bootloader-theme>SLE</bootloader-theme>

        <locale>en_US</locale>
        <keytable>us.map.gz</keytable>
        <timezone>Europe/Berlin</timezone>
        <hwclock>utc</hwclock>

        <rpm-excludedocs>true</rpm-excludedocs>
        <type boot="saltboot/suse-SLES12" bootloader="grub2" checkprebuilt="true"
compressed="false" filesystem="ext3" fsmountoptions="acl" fsnocheck="true" image="pxe"
kernelcmdline="quiet"></type>
    </preferences>
    <!--    CUSTOM REPOSITORY
    <repository type="rpm-dir">
      <source path="this://repo"/>
    </repository>
    -->
    <packages type="image">
        <package name="patterns-sles-Minimal"/>
        <package name="aaa_base-extras"/> <!-- wouldn't be SUSE without that ;-) -->
        <package name="kernel-default"/>
        <package name="salt-minion"/>
        ...
    </packages>
    <packages type="bootstrap">
        ...
        <package name="sles-release"/>
        <!-- this certificate package is required to access {productname} repositories
             and is provided by {productname} automatically -->
        <package name="rhn-org-trusted-ssl-cert-osimage" bootinclude="true"/>

    </packages>
    <packages type="delete">
        <package name="mtools"/>
        <package name="initviocons"/>
        ...
    </packages>
</image>
```

Build an Image

There are two ways to build an image using the {webui}. Either select menu:Images[Build], or click the build icon in the menu:Images[Profiles] list.

Build Image ⚙

Version:	latest
Image Profile *:	Select an image profile ▾
Build Host *:	Select a build host ▾
⊙Earliest:	📅 05.06.` 🕐 18:0 CEST
⚪Add to:	new action chain ▾
	⚙ Build

Profile Summary

ⓘ No profile selected

Procedure: Building an Image

1. Select menu:Images[Build].

2. Add a different tag name if you want a version other than the default `latest` (applies only to containers).

3. Select the `Image Profile` and a `Build Host`.

   > **NOTE**
   > A `Profile Summary` is displayed to the right of the build fields. When you have selected a build profile, detailed information about the selected profile is shown here.

4. To schedule a build, click the btn:[Build] button.

   > **IMPORTANT**
   > The build server cannot run any form of automounter during the image building process. If applicable, ensure that you do not have your Gnome session running as root. If an automounter is running, the image build finishes successfully, but the checksum of the image is different and causes a failure.

After the image is successfully built, the inspection phase begins. During the inspection phase {susemgr} collects information about the image:

- List of packages installed in the image

- Checksum of the image

- Image type and other image details

> **NOTE**
> If the built image type is `PXE`, a Salt pillar is also generated. Image pillars are stored in the `/srv/susemanager/pillar_data/images/` directory and the Salt subsystem can access details about the generated image. Details include where the pillar is located and provided, image checksums, information needed for network boot, and more.
>
> The generated pillar is available to all connected clients.

Troubleshooting

Building an image requires several dependent steps. When the build fails, investigating Salt states results can help identify the source of the failure. You can carry out these checks when the build fails:

- The build host can access the build sources

- There is enough disk space for the image on both the build host and the Uyuni server

- The activation key has the correct channels associated with it

- The build sources used are valid

- The RPM package with the Uyuni public certificate is up to date and available at `/usr/share/susemanager/salt/images/rhn-org-trusted-ssl-cert-osimage-1.0-1.noarch.rpm`. For more on how to refresh a public certificate RPM, see Create a Build Host.

Limitations

The section contains some known issues when working with images.

- HTTPS certificates used to access the HTTP sources or git repositories should be deployed to the client by a custom state file, or configured manually.

- Importing Kiwi-based images is not supported.

List Image Profiles Available for Building

To list images available for building select menu:Images[Image List]. A list of all images is displayed.
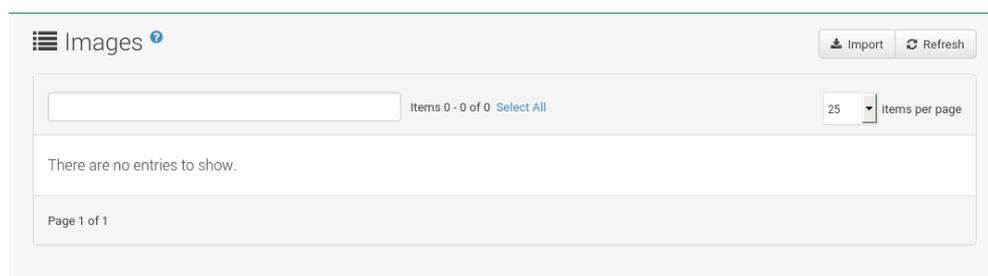
Displayed data about images includes an image `Name`, its `Version` and the build `Status`. You can also see the image update status with a listing of possible patch and package updates that are available for the image.

Clicking the btn:[Details] button on an image provides a detailed view. The detailed view includes an exact list of relevant patches and a list of all packages installed within the image.

> **NOTE** The patch and the package list is only available if the inspect state after a build was successful.

Channel Management

Channels are a method of grouping software packages.

In Uyuni, channels are grouped into base and child channels, with base channels grouped by operating system type, version, and architecture, and child channels being compatible with their related base channel. When a client has been assigned to a base channel, it is only possible for that system to install the related child channels. Organizing channels in this way ensures that only compatible packages are installed on each system.

Software channels use repositories to provide packages. The channels mirror the repositories in Uyuni, and the package names and other data are stored in the Uyuni database. You can have any number of repositories associated with a channel. The software from those repositories can then be installed on clients by subscribing the client to the appropriate channel.

Clients can only be assigned to one base channel. The client can then install or update packages from the repositories associated with that base channel and any of its child channels.

Uyuni provides a number of vendor channels, which provide you everything you need to run Uyuni. Uyuni Administrators and Channel Administrators have channel management authority, which gives them the ability to create and manage their own custom channels. If you want to use your own packages in your environment, you can create custom channels. Custom channels can be used as a base channel, or you can associate them with a vendor base channel.

For more on creating custom channels, see **[Administration › Custom-channels ›]**.

Channel Administration

By default, any user can subscribe channels to a system. You can implement restrictions on the channel using the {webui}.

Procedure: Restricting Subscriber Access to a Channel

1. In the Uyuni {webui}, navigate to menu:Software[Channel List], and select the channel to edit.

2. Locate the `Per-User Subscription Restrictions` section and check `Only selected users within your organization may subscribe to this channel`. Click btn:[Update] to save the changes.

3. Navigate to the `Subscribers` tab and select or deselect users as required.

Delete Channels

You can delete vendor software channels from the command prompt.

For information about deleting custom channels, see **[Administration › Custom-channels ›]**.

Procedure: Deleting Vendor Channels

1. On the Uyuni Server, at the command prompt, as root, list the available vendor channels, and make a note of the channel you want to delete:

```
mgr-sync list channels
```

2. Delete the channel:

```
spacewalk-remove-channel <channel-name>
```

3. Synchronize to remove the channel:

```
mgr-sync sync channel <channel-name>
```

Custom Channels

Custom channels give you the ability to create your own software packages and repositories, which you can use to update your clients. They also allow you to use software provided by third party vendors in your environment.

You must have administrator privileges to be able to create and manage custom channels.

Before you create a custom channel, determine which base channel you want to associate it with, and which repositories you want to use for content.

This section gives more detail on how to create, administer, and delete custom channels.

Creating Custom Channels and Repositories

If you have custom software packages that you need to install on your Uyuni systems, you can create a custom child channel to manage them. You need to create the channel in the Uyuni {webui} and create a repository for the packages, before assigning the channel to your systems.

You can select a vendor channel as the base channel if you want to use packages provided by a vendor. Alternatively, select `none` to make your custom channel a base channel.

Custom channels sometimes require additional security settings. Many third party vendors secure packages with GPG. If you want to use GPG-protected packages in your custom channel, you need to trust the GPG key which has been used to sign the metadata. You can then check the `Has Signed Metadata?` check box to match the package metadata against the trusted GPG keys. For more information on importing GPG keys, see **[Reference › Systems ›]**.

IMPORTANT	Do not create child channels containing packages that are not compatible with the client system.

Procedure: Creating a Custom Channel

1. In the Uyuni {webui}, navigate to menu:Software[Manage > Channels], and click btn:[Create Channel].

2. On the `Create Software Channel` page, give your channel a name (for example, `My Tools SLES 15 SP1 x86_64`) and a label (for example, `my-tools-sles15sp1-x86_64`). Labels must not contain spaces or uppercase letters.

3. In the `Parent Channel` drop down, choose the relevant base channel (for example, `SLE-Product-SLES15-SP1-Pool for x86_64`). Ensure that you choose the compatible parent channel for your packages.

4. In the `Architecture` drop down, choose the appropriate hardware architecture (for example, `x86_64`).

5. Provide any additional information in the contact details, channel access control, and GPG fields, as required for your environment.

6. Click btn:[Create Channel].

IMPORTANT	By default, the `Enable GPG Check` field is checked when you create a new channel. If you would like to add custom packages and applications to your channel, make sure you uncheck this field to be able to install unsigned packages. Disabling the GPG check is a security risk if packages are from an untrusted source.

NOTE	If you are registering traditional {rhel} 7 or SLES {es} 7 clients you might notice errors with unsigned packages. For more information, see [**Administration › Tshoot-registertradrhel › **].

Procedure: Creating a Software Repository

1. In the Uyuni {webui}, navigate to menu:Software[Manage > Repositories], and click btn:[Create Repository].

2. On the `Create Repository` page, give your repository a label (for example, `my-tools-sles15sp1-x86_64-repo`).

3. In the `Repository URL` field, provide the path to the directory that contains the `repodata` file (for example, `file:///opt/mytools/`). You can use any valid addressing protocol in this field.

4. Uncheck the `Has Signed Metadata?` check box.

5. OPTIONAL: Complete the SSL fields if your repository requires client certificate authentication.

6. Click btn:[Create Repository].

Procedure: Assigning the Repository to a Channel

1. Assign your new repository to your custom channel by navigating to menu:Software[Manage > Channels], clicking the name of your newly created custom channel, and navigating to the `Repositories` tab.

2. Ensure the repository you want to assign to the channel is checked, and click btn:[Update Repositories].

3. Navigate to the `Sync` tab and click btn:[Sync Now] to synchronize immediately. You can also set an automated synchronization schedule on this tab.

There are several ways to check if a channel has finished synchronizing:

- In the Uyuni {webui}, navigate to menu:Admin[Setup Wizard] and select the `Products` tab. This dialog displays a completion bar for each product when they are being synchronized.

- In the Uyuni {webui}, navigate to menu:Software[Manage > Channels], then click the channel associated to the repository. Navigate to the menu:[Repositories > Sync] tab. The `Sync Status` is shown next to the repository name..

- Check the synchronization log file at the command prompt:

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

Each child channel generates its own log during the synchronization progress. You need to check all the base and child channel log files to be sure that the synchronization is complete.

Procedure: Adding Custom Channels to an Activation Key

1. In the Uyuni {webui}, navigate to menu:Systems[Activation Keys], and select the key you want to add the custom channel to.

2. On the `Details` tab, in the `Child Channels` listing, select the channel to associate. You can select multiple channels, if you need to.

3. Click btn:[Update Activation Key].

Add Packages and Patches to Custom Channels

When you create a new custom channel without cloning it from an existing channel, it does not contain any packages or patches. You can add the packages and patches you require using the Uyuni {webui}.

Custom channels can only include packages or patches that are cloned or custom, and they must match the base architecture of the channel. Patches added to custom channels must apply to a package that exists in the channel.

Procedure: Adding Packages to Custom Channels

1. In the Uyuni {webui}, navigate to menu:Software[Manage > Channels], and go to the `Packages` tab.

2. OPTIONAL: See all packages currently in the channel by navigating to the `List/Remove` tab.

3. Add new packages to the channel by navigating to the `Add` tab.

4. Select the parent channel to provide packages, and click btn:[View Packages] to populate the list.

5. Check the packages to add to the custom channel, and click btn:[Add Packages].

6. When you are satisfied with the selection, click btn:[Confirm Addition] to add the packages to the channel.

7. OPTIONAL: You can compare the packages in the current channel with those in a different channel by navigating to menu:Software[Manage > Channels], and going to the menu:Packages[Compare] tab. To make the two channels the same, click the btn:[Merge Differences] button, and resolve any conflicts.

Procedure: Adding Patches to a Custom Channel

1. In the Uyuni {webui}, navigate to menu:Software[Manage > Channels], and go to the `Patches` tab.

2. OPTIONAL: See all patches currently in the channel by navigating to the `List/Remove` tab.

3. Add new patches to the channel by navigating to the `Add` tab, and selecting what kind of patches you want to add.

4. Select the parent channel to provide patches, and click btn:[View Associated Patches] to populate the list.

5. Check the patches to add to the custom channel, and click btn:[Confirm].

6. When you are satisfied with the selection, click btn:[Confirm] to add the patches to the channel.

Manage Custom Channels

Uyuni administrators and channel administrators can alter or delete any channel.

To grant other users rights to alter or delete a channel, navigate to menu:Software[Manage > Channels] and select the channel you want to edit. Navigate to the `Managers` tab, and check the user to grant permissions. Click btn:[Update] to save the changes.

> **IMPORTANT**
>
> If you delete a channel that has been assigned to a set of clients, it triggers an immediate update of the channel state for any clients associated with the deleted channel. This is to ensure that the changes are reflected accurately in the repository file.

You cannot delete Uyuni channels with the {webui}. Only custom channels can be deleted.

Procedure: Deleting Custom Channels

1. In the Uyuni {webui}, navigate to menu:Software[Manage > Channels], and select the channel you want to delete.

2. Click btn:[Delete software channel].

3. On the `Delete Channel` page, check the details of the channel you are deleting, and check the `Unsubscribe Systems` checkbox to remove the custom channel from any systems that might still be subscribed.

4. Click btn:[Delete Channel].

When channels are deleted, the packages that are part of the deleted channel are not automatically removed. You are not able to update packages that have had their channel deleted.

You can delete packages that are not associated with a channel in the Uyuni {webui}. Navigate to menu:Software[Manage > Packages], check the packages to remove, and click btn:[Delete Packages].

Patch Management

This chapter contains various topics related to patch management.

Retracted Patches

When a new patch gets released by the vendor, it might happen that the patch has undesirable side effects (security, stability) in some scenario that was not identified by testing. When this happens (very rarely), vendors typically release a new patch, which may take hours or days, depending on the internal processes in place by that vendor.

SUSE has introduced a new mechanism (2021) called "retracted patches" to revoke such patches almost immediately by setting their advisory status to "retracted" (instead of "final" or "stable").

Definitions: A patch is "retracted", when its advisory status attribute is set to "retracted". A package is "retracted", when it belongs to a "retracted" patch.

A retracted patch or package cannot be installed on systems with Uyuni. The only way to install a retracted package, is to do it manually with `zypper install` and specifying the exact package version. For example:

```
zypper install vim-8.0.1568-5.14.1
```

Retracted status of patches and packages is depicted with the [times circle] icon in the {webui} of Uyuni. For example, see: * list of packages in a channel * list of patches in a channel

When a patch or package, that has been installed on a system, gets retracted, the [times circle] icon is also displayed in the installed packages list of that system. Uyuni does not provide a way to downgrade such a patch or package.

Channel Clones

When using cloned channels, you must pay attention to the propagation of the retracted advisory status from the original channels to the clones.

Upon cloning vendor channels into your organization, channel patches will be cloned as well.

When the vendor retracts a patch in a channel and Uyuni synchronizes this channel (for example, with the nightly job), the "retracted" attribute will not get propagated to the cloned patches and will not be

observed by the clients subscribed to cloned channels. To propagate the attribute to your cloned channels use the following ways:

- Patch Sync (menu:Software[Manage > cloned channel > Patches > Sync]). This function allows you to align the attributes of patches in your cloned channel to their originals.

- Content Lifecycle Management. For more information about cloned channels in the context of Content Lifecycle Management, see **[Client-configuration › Channels ›]**.

Patch sharing

When you create multiple vendor channel clones in your organization, the patches are not cloned multiple times, but are shared between cloned channels. As a consequence, when you synchronize your cloned patch (either using the patch sync function or with Content Lifecycle Management mentioned above), all channels using the cloned patch will observe that change.

Example

1. Consider two Content Lifecycle Management projects `prj1` and `prj2`

2. Both of these projects have 2 environments `dev` and `test`

3. Both of these projects have a vendor channel set as a source channel

4. All channels in this scenario (four cloned channels in total) are aligned to the latest state of the vendor channels

5. Vendor retracts a patch in the source channel and the nighly job synchronizes it to your Uyuni

6. None of the four channels see this change because they are using a patch clone, not the patch directly.

7. As soon as you synchronize your patch (either you build any of these two projects, or you use the Patch Sync function on any of the four cloned channels), due to the patch sharing, **ALL** of the cloned channels will see the patch as retracted.

Subscription Matching

Your {suse} products require subscriptions, which are managed by the {scc} (SCC). Uyuni runs a nightly report checking the subscription status of all your registered clients against your SCC account. The report gives you information about which clients consume which subscriptions, how many subscriptions you have remaining and available to use, and which clients do not have a current subscription.

Navigate to menu:Audit[Subscription Matching] to see the report.

The `Subscriptions Report` tab gives information about current and expiring subscriptions.

The `Unmatched Products Report` tab gives a list of clients that do not have a current subscription. This includes clients that could not be matched, or that are not currently registered with Uyuni. The report includes product names and the number of systems that remain unmatched.

The `Pins` tab allows you to associate individual clients to the relevant subscription. This is especially useful if the subscription manager is not automatically associating clients to subscriptions successfully.

The `Messages` tab shows any errors that occurred during the matching process.

You can also download the reports in .csv format, or access them from that command prompt in the `/var/lib/spacewalk/subscription-matcher/` directory.

By default, the subscription matcher runs daily, at midnight. To change this, navigate to menu:Admin[Task Schedules] and click `gatherer-matcher-default`. Change the schedule as required, and click btn:[Update Schedule].

Because the report can only match current clients with current subscriptions, you might find that the matches change over time. The same client does not always match the same subscription. This can be due to new clients being registered or unregistered, or because of the addition or expiration of subscriptions.

The subscription matcher automatically attempts to reduce the number of unmatched products, limited by the terms and conditions of the subscriptions in your account. However, if you have incomplete hardware information, unknown virtual machine host assignments, or clients running in unknown public clouds, the matcher might show that you do not have enough subscriptions available. Always ensure you have complete data about your clients included in Uyuni, to help ensure accuracy.

IMPORTANT
The subscription matcher does not always match clients and subscriptions accurately. It is not intended to be a replacement for auditing.

Pin Clients to Subscriptions

If the subscription matcher is not automatically matching a particular client with the correct subscription, you can manually pin them. When you have created a pin, the subscription matcher favors matching a specific subscription with a given system or group of systems.

However, the matcher does not always respect a pin. It depends on the subscription being available, and whether or not the subscription can be applied to the client. Additionally, pins are ignored if they result in a match that violates the terms and conditions of the subscription, or if the matcher detects a more accurate match if the pin is ignored.

To add a new pin, click btn:[Add a Pin], and select the client to pin.

IMPORTANT
We do not recommend using pinning regularly, or for a large number of clients. The subscription matcher tool is generally accurate enough for most installations.

Live Patching with SUSE Manager

Performing a kernel update usually requires a system reboot. Common vulnerability and exposure (CVE) patches should be applied as soon as possible, but if you cannot afford the downtime, you can use Live

Patching to inject these important updates and skip the need to reboot.

The procedure for setting up Live Patching is slightly different for SLES 12 and SLES 15. Both procedures are documented in this section.

Set up Channels for Live Patching

A reboot is required every time you update the full kernel package. Therefore, it is important that clients using Live Patching do not have newer kernels available in the channels they are assigned to. Clients using live patching have updates for the running kernel in the live patching channels.

There are two ways to manage channels for live patching:

Use content lifecycle management to clone the product tree and remove kernel versions newer than the running one. This procedure is explained in the content-lifecycle-examples.pdf. This is the recommended solution.

Alternatively, use the `spacewalk-manage-channel-lifecycle` tool. This procedure is more manual and requires command line tools as well as the {webui}. This procedure is explained in this section for SLES 15 SP1, but it also works for SLE 12 SP4 or later.

Use spacewalk-manage-channel-lifecycle for Live Patching

Cloned vendor channels should be prefixed by `dev` for development, `testing`, or `prod` for production. In this procedure, you create a `dev` cloned channel and then promote the channel to `testing`.

Procedure: Cloning Live Patching Channels

1. At the command prompt on the client, as root, obtain the current package channel tree:

   ```
   # spacewalk-manage-channel-lifecycle --list-channels
   Spacewalk Username: admin
   Spacewalk Password:
   Channel tree:

    1. sles15-{sp-vert}-pool-x86_64
          \__ sle-live-patching15-pool-x86_64-{sp-vert}
          \__ sle-live-patching15-updates-x86_64-{sp-vert}
          \__ sle-manager-tools15-pool-x86_64-{sp-vert}
          \__ sle-manager-tools15-updates-x86_64-{sp-vert}
          \__ sles15-{sp-vert}-updates-x86_64
   ```

2. Use the `spacewalk-manage-channel` command with the `init` argument to automatically create a new development clone of the original vendor channel:

   ```
   spacewalk-manage-channel-lifecycle --init -c sles15-{sp-vert}-pool-x86_64
   ```

3. Check that `dev-sles15-{sp-vert}-updates-x86_64` is available in your channel list.

Check the `dev` cloned channel you created, and remove any kernel updates that require a reboot.

Procedure: Removing Non-Live Kernel Patches from Cloned Channels

1. Check the current kernel version by selecting the client from menu:Systems[System List], and taking note of the version displayed in the `Kernel` field.

2. In the Uyuni {webui}, select the client from menu:Systems[Overview], navigate to the menu:Software[Manage > Channels] tab, and select `dev-sles15-{sp-vert}-updates-x86_64`. Navigate to the `Patches` tab, and click btn:[List/Remove Patches].

3. In the search bar, type `kernel` and identify the kernel version that matches the kernel currently used by your client.

4. Remove all kernel versions that are newer than the currently installed kernel.

Your channel is now set up for live patching, and can be promoted to `testing`. In this procedure, you also add the live patching child channels to your client, ready to be applied.

Procedure: Promoting Live Patching Channels

1. At the command prompt on the client, as `root`, promote and clone the `dev-sles15-{sp-vert}-pool-x86_64` channel to a new `testing` channel:

   ```
   # spacewalk-manage-channel-lifecycle --promote -c dev-sles15-{sp-vert}-pool-x86_64
   ```

2. In the Uyuni {webui}, select the client from menu:Systems[Overview], and navigate to the menu:Software[Software Channels] tab.

3. Check the new `test-sles15-{sp-vert}-pool-x86_64` custom channel to change the base channel, and check both corresponding live patching child channels.

4. Click btn:[Next], confirm that the details are correct, and click btn:[Confirm] to save the changes.

You can now select and view available CVE patches, and apply these important kernel updates with Live Patching.

Live Patching on SLES 15

On SLES 15 systems and newer, live patching is managed by the `klp livepatch` tool.

Before you begin, ensure:

- Uyuni is fully updated.

- You have one or more Salt clients running SLES 15 (SP1 or later).

- Your SLES 15 Salt clients are registered with Uyuni.

- You have access to the SLES 15 channels appropriate for your architecture, including the live patching child channel (or channels).

- The clients are fully synchronized.

- Assign the clients to the cloned channels prepared for live patching. For more information on preparation, see **[Administration › Live-patching-channel-setup ›]**.

Procedure: Setting up for Live Patching

1. Select the client you want to manage with Live Patching from menu:Systems[Overview], and navigate to the menu:Software[Packages > Install] tab. Search for the `kernel-livepatch` package, and install it.

2. Apply the highstate to enable Live Patching, and reboot the client.

3. Repeat for each client that you want to manage with Live Patching.

4. To check that live patching has been enabled correctly, select the client from menu:Systems[System List], and ensure that `Live Patch` appears in the `Kernel` field.

Procedure: Applying Live Patches to a Kernel

1. In the Uyuni {webui}, select the client from menu:Systems[Overview]. A banner at the top of the screen shows the number of critical and non-critical packages available for the client:

2. Click btn:[Critical] to see a list of the available critical patches.

3. Select any patch with a synopsis reading `Important: Security update for the Linux kernel`. Security bugs also include their CVE number, where applicable.

4. OPTIONAL: If you know the CVE number of a patch you want to apply, you can search for it in menu:Audit[CVE Audit], and apply the patch to any clients that require it.

IMPORTANT	Not all kernel patches are Live Patches. Non-Live kernel patches are represented by a `Reboot Required` icon located next to the `Security` shield icon. These patches always require a reboot.

IMPORTANT	Not all security issues can be fixed by applying a live patch. Some security issues can only be fixed by applying a full kernel update and requires a reboot. The assigned CVE numbers for these issues are not included in live patches. A CVE audit displays this requirement.

Live Patching on SLES 12

On SLES 12 systems, live patching is managed by kGraft. For in depth information covering kGraft use, see https://documentation.suse.com/sles/12-SP4/html/SLES-all-cha-kgraft.html.

Before you begin, ensure:

- Uyuni is fully updated.

- You have one or more Salt clients running SLES 12 (SP1 or later).

- Your SLES 12 Salt clients are registered with Uyuni.

- You have access to the SLES 12 channels appropriate for your architecture, including the live patching child channel (or channels).

- The clients are fully synchronized.

- Assign the clients to the cloned channels prepared for live patching. For more information on preparation, see **[Administration › Live-patching-channel-setup ›]**.

Procedure: Setting up for Live Patching

1. Select the client you want to manage with Live Patching from menu:Systems[Overview], and on the system details page navigate to the menu:Software[Packages > Install] tab. Search for the `kgraft` package, and install it.
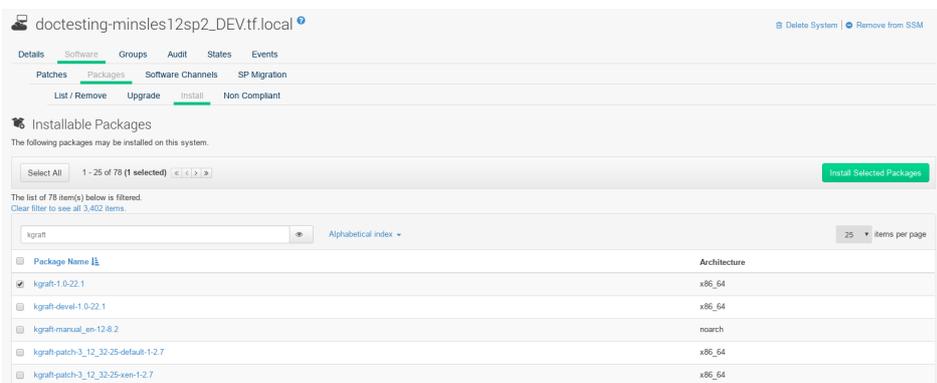
2. Apply the highstate to enable Live Patching, and reboot the client.

3. Repeat for each client that you want to manage with Live Patching.

4. To check that live patching has been enabled correctly, select the client from menu:Systems[System List], and ensure that `Live Patching` appears in the `Kernel` field.

Procedure: Applying Live Patches to a Kernel

1. In the Uyuni {webui}, select the client from menu:Systems[Overview]. A banner at the top of the screen shows the number of critical and non-critical packages available for the client:

System Status

❗ Software Updates Available   Critical: 1   Non-Critical: 2   Packages: 3

2. Click btn:[Critical] to see a list of the available critical patches.

3. Select any patch with a synopsis reading `Important: Security update for the Linux kernel`. Security bugs also include their CVE number, where applicable.

4. OPTIONAL: If you know the CVE number of a patch you want to apply, you can search for it in menu:Audit[CVE Audit], and apply the patch to any clients that require it.

IMPORTANT	Not all kernel patches are Live Patches. Non-Live kernel patches are represented by a `Reboot Required` icon located next to the `Security` shield icon. These patches always require a reboot.
IMPORTANT	Not all security issues can be fixed by applying a live patch. Some security issues can only be fixed by applying a full kernel update and require a reboot. The assigned CVE numbers for these issues are not included in live patches. A CVE audit displays this requirement.

Monitoring with Prometheus and Grafana

You can monitor your Uyuni environment using Prometheus and Grafana. Uyuni Server and Proxy are able to provide self-health metrics. You can also install and manage a number of Prometheus exporters on Salt clients.

Prometheus and Grafana packages are included in the Uyuni Client Tools for:

- {sle} 12
- {sle} 15
- {centos} 6
- {centos} 7

- {centos} 8

- and openSUSE 15.x

You need to install Prometheus and Grafana on a machine separate from the Uyuni Server. We recommend you use a managed Salt client as your monitoring server.

Prometheus fetches metrics using a pull mechanism, so the server must be able to establish TCP connections to monitored clients. Clients must have corresponding open ports and be reachable over the network. Alternatively, you can use reverse proxies to establish a connection.

Prometheus and Grafana

Prometheus

Prometheus is an open-source monitoring tool that is used to record real-time metrics in a time-series database. Metrics are pulled via HTTP, enabling high performance and scalability.

Prometheus metrics are time series data, or timestamped values belonging to the same group or dimension. A metric is uniquely identified by its name and set of labels.

```
    metric name              labels               timestamp     value

 http_requests_total{status="200", method="GET"}  @1557331801.111  42236
```
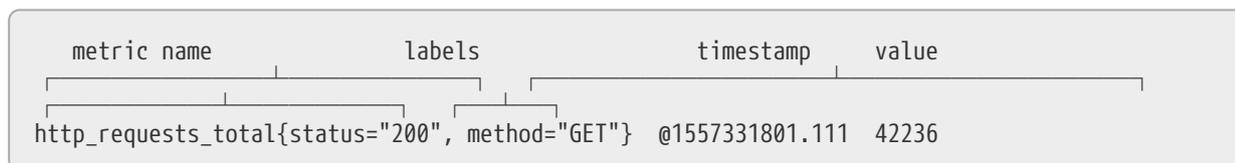
Each application or system being monitored must expose metrics in the format above, either through code instrumentation or Prometheus exporters.

Prometheus Exporters

Exporters are libraries that help with exporting metrics from third-party systems as Prometheus metrics. Exporters are useful whenever it is not feasible to instrument a given application or system with Prometheus metrics directly. Multiple exporters can run on a monitored host to export local metrics.

The Prometheus community provides a list of official exporters, and more can be found as community contributions. For more information and an extensive list of exporters, see https://prometheus.io/docs/instrumenting/exporters/.

Grafana

Grafana is a tool for data visualization, monitoring, and analysis. It is used to create dashboards with panels representing specific metrics over a set period of time. Grafana is commonly used together with Prometheus, but also supports other data sources such as ElasticSearch, MySQL, PostgreSQL, and Influx DB. For more information about Grafana, see https://grafana.com/docs/.

Set up the Monitoring Server

To set up your monitoring server, you need to install Prometheus and Grafana, and configure them.

Install Prometheus

If your monitoring server is a Salt client, you can install the Prometheus package using the Uyuni {webui}. Otherwise you can download and install the package on your monitoring server manually. The Prometheus software is also available for Uyuni Proxy and Uyuni {smr} Branch Server.

> **NOTE** Prometheus expects POSIX filesystem for storing data. Non-POSIX compliant filesystems are not supported. NFS filesystems are explicitly not supported.

Procedure: Installing Prometheus Using the {webui}

1. In the Uyuni {webui}, open the details page of the system where Prometheus is to be installed, and navigate to the `Formulas` tab.

2. Check the `Prometheus` checkbox to enable monitoring formulas, and click btn:[Save].

3. Navigate to the `Prometheus` tab in the top menu.

4. In the `Uyuni Server` section, enter valid Uyuni API credentials. Make sure that the credentials you have entered allow access to the set of systems you want to monitor.

5. Customize any other configuration options according to your needs.

6. Click btn:[Save Formula].

7. Apply the highstate and confirm that it completes successfully.

8. Check that the Prometheus interface loads correctly. In your browser, navigate to the URL of the server where Prometheus is installed, on port 9090 (for example, `http://example.com:9090`).

For more information about the monitoring formulas, see **[Salt › Formula-monitoring ›]**.

Procedure: Manually Installing and Configuring Prometheus

1. On the monitoring server, install the `golang-github-prometheus-prometheus` package:

   ```
   zypper in golang-github-prometheus-prometheus
   ```

2. Enable the Prometheus service:

   ```
   systemctl enable --now prometheus
   ```

3. Check that the Prometheus interface loads correctly. In your browser, navigate to the URL of the server where Prometheus is installed, on port 9090 (for example, `http://example.com:9090`).

4. Open the configuration file at `/etc/prometheus/prometheus.yml` and add this configuration information. Replace `server.url` with your Uyuni server URL and adjust `username` and `password` fields to match your Uyuni credentials.

```
# {productname} self-health metrics
scrape_configs:
- job_name: 'mgr-server'
  static_configs:
    - targets:
      - 'server.url:9100'  # Node exporter
      - 'server.url:9187'  # PostgreSQL exporter
      - 'server.url:5556'  # JMX exporter (Tomcat)
      - 'server.url:5557'  # JMX exporter (Taskomatic)
      - 'server.url:9800'  # Taskomatic
    - targets:
      - 'server.url:80'    # Message queue
      labels:
        __metrics_path__: /rhn/metrics

# Managed systems metrics:
- job_name: 'mgr-clients'
  uyuni_sd_configs:
    - host: "http://server.url"
      username: "admin"
      password: "admin"
```

5. Save the configuration file.

6. Restart the Prometheus service:

```
systemctl restart prometheus
```

For more information about the Prometheus configuration options, see the official Prometheus documentation at https://prometheus.io/docs/prometheus/latest/configuration/configuration/

Install Grafana

If your monitoring server is a Salt client, you can install the Grafana package using the Uyuni {webui}. Otherwise you can download and install the package on your monitoring server manually.

Procedure: Installing Grafana Using the {webui}

1. In the Uyuni {webui}, open the details page of the system where Grafana is to be installed, and navigate to the `Formulas` tab.

2. Check the `Grafana` checkbox to enable monitoring formulas, and click btn:[Save].

3. Navigate to the `Grafana` tab in the top menu.

4. In the `Enable and configure Grafana` section, enter the admin credentials you want to use to log in Grafana.

5. On the `Datasources` section, make sure that the Prometheus URL field points to the system where Prometheus is running.

6. Customize any other configuration options according to your needs.

7. Click btn:[Save Formula].

8. Apply the highstate and confirm that it completes successfully.

9. Check that the Grafana interface is loading correctly. In your browser, navigate to the URL of the server where Grafana is installed, on port 3000 (for example, `http://example.com:3000`).

> **NOTE**
>
> Uyuni provides pre-built dashboards for server self-health, basic client monitoring, and more. You can choose which dashboards to provision in the formula configuration page.

For more information about the monitoring formulas, see **[Salt › Formula-monitoring ›]**.

Procedure: Manually Installing Grafana

1. Install the `grafana` package:

```
zypper in grafana
```

2. Enable the Grafana service:

```
systemctl enable --now grafana-server
```

3. Check that the Grafana interface is loading correctly. In your browser, navigate to the URL of the server where Grafana is installed, on port 3000 (for example, `http://example.com:3000`).
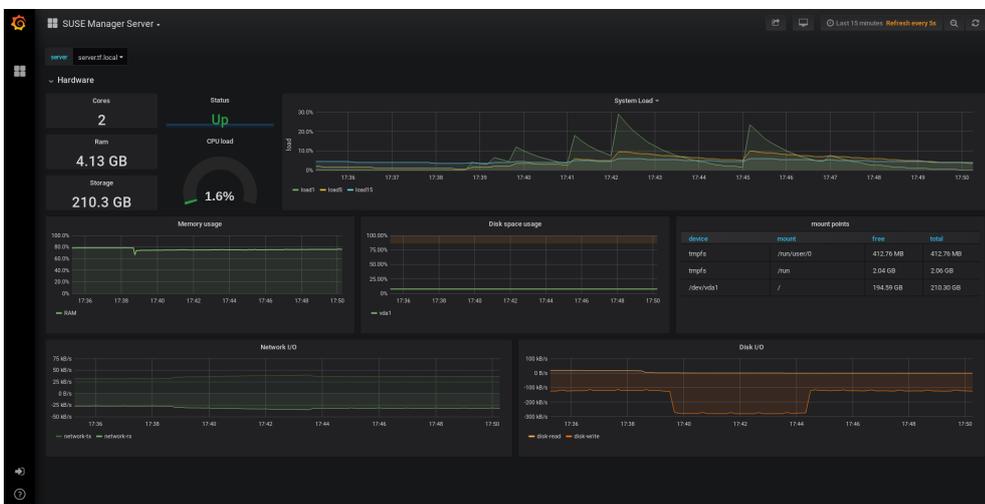
For more information on how to manually install and configure Grafana, see https://grafana.com/docs.

For more information about the monitoring formulas with forms, see **[Salt › Formula-monitoring ›]**.

Configure Uyuni Monitoring

With Uyuni 4 and higher, you can enable the server to expose Prometheus self-health metrics, and also install and configure exporters on client systems.

Server Self Monitoring

The Server self-health metrics cover hardware, operating system and Uyuni internals. These metrics are made available by instrumentation of the Java application, combined with Prometheus exporters.

These exporter packages are shipped with Uyuni Server:

- Node exporter: `golang-github-prometheus-node_exporter`. See https://github.com/prometheus/node_exporter.

- PostgreSQL exporter: `golang-github-wrouesnel-postgres_exporter`. See https://github.com/wrouesnel/postgres_exporter.

- JMX exporter: `prometheus-jmx_exporter`. See https://github.com/prometheus/jmx_exporter.

- Apache exporter: `golang-github-lusitaniae-apache_exporter`. See https://github.com/Lusitaniae/apache_exporter.

These exporter packages are shipped with Uyuni Proxy:

- Node exporter: `golang-github-prometheus-node_exporter`. See https://github.com/prometheus/node_exporter.

- Squid exporter: `golang-github-boynux-squid_exporter`. See https://github.com/boynux/squid-exporter.

The exporter packages are pre-installed in Uyuni Server and Proxy, but their respective systemd daemons are disabled by default.

Procedure: Enabling Self Monitoring

1. In the Uyuni {webui}, navigate to menu:Admin[Manager Configuration > Monitoring].

2. Click btn:[Enable services].

3. Restart Tomcat and Taskomatic.

4. Navigate to the URL of your Prometheus server, on port 9090 (for example, `http://example.com:9090`)

5. In the Prometheus UI, navigate to menu:[Status > Targets] and confirm that all the endpoints on the `mgr-server` group are up.

6. If you have also installed Grafana with the {webui}, the server insights are visible on the Uyuni Server dashboard.
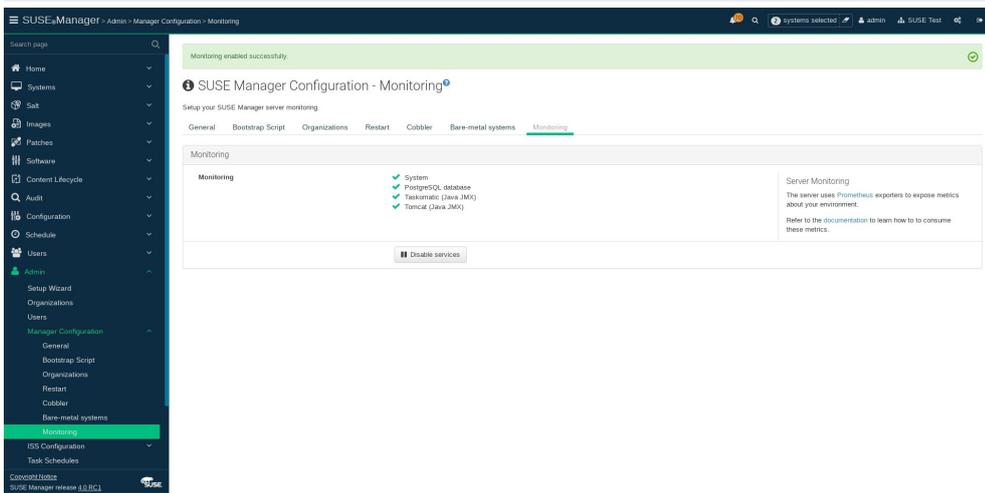
IMPORTANT

Only server self-health monitoring can be enabled using the {webui}. Metrics for a proxy are not automatically collected by Prometheus. To enable self-health monitoring on a proxy, you need to manually install exporters and enable them.

Monitoring Managed Systems

Prometheus metrics exporters can be installed and configured on Salt clients using formulas. The packages are available from the Uyuni client tools channels, and can be enabled and configured directly in the Uyuni {webui}.

These exporters can be installed on managed systems:

- Node exporter: `golang-github-prometheus-node_exporter`. See https://github.com/prometheus/node_exporter.

- PostgreSQL exporter: `golang-github-wrouesnel-postgres_exporter`. See https://github.com/wrouesnel/postgres_exporter.

- Apache exporter: `golang-github-lusitaniae-apache_exporter`. See https://github.com/Lusitaniae/apache_exporter.

When you have the exporters installed and configured, you can start using Prometheus to collect metrics from monitored systems. If you have configured your monitoring server with the {webui}, metrics collection happens automatically.

Procedure: Configuring Prometheus Exporters on a Client

1. In the Uyuni {webui}, open the details page of the client to be monitored, and navigate to the menu:Formulas tab.

2. Check the `Enabled` checkbox on the `Prometheus Exporters` formula.

3. Click btn:[Save].

4. Navigate to the menu:Formulas[Prometheus Exporters] tab.

5. Select the exporters you want to enable and customize arguments according to your needs. The `Address` field accepts either a port number preceded by a colon (`:9100`), or a fully resolvable address (`example:9100`).

6. Click btn:[Save Formula].

7. Apply the highstate.
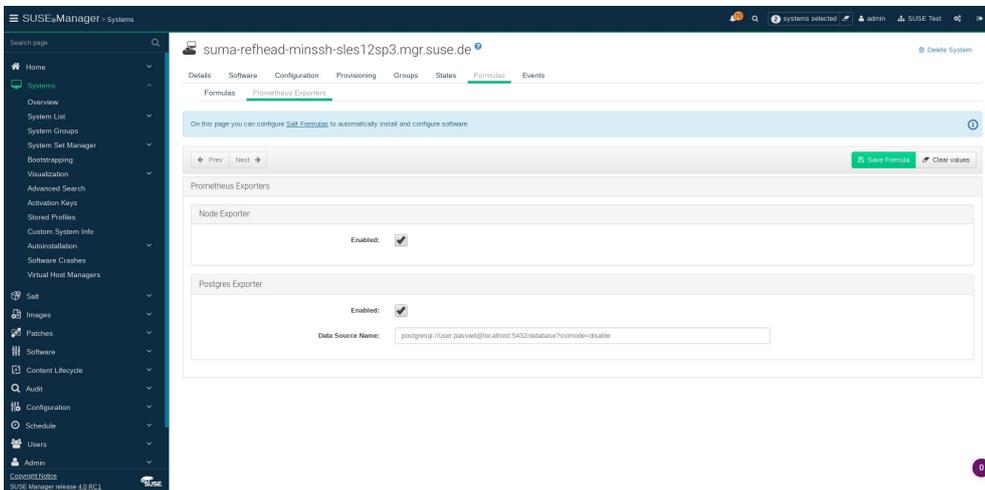
> **NOTE**
>
> Monitoring formulas can also be configured for System Groups, by applying the same configuration used for individual systems inside the corresponding group.

For more information about the monitoring formulas, see **[Salt › Formula-monitoring ›]**.

Network Boundaries

Prometheus fetches metrics using a pull mechanism, so the server must be able to establish TCP connections to monitored clients. By default, Prometheus uses these ports:

- Node exporter: 9100

- PostgreSQL exporter: 9187

- Apache exporter: 9117

Additionally, if you are running the alert manager on a different host than where you run Prometheus, you also need to open port 9093.

For clients installed on cloud instances, you can add the required ports to a security group that has access to the monitoring server.

Alternatively, you can deploy a Prometheus instance in the exporters' local network, and configure federation. This allows the main monitoring server to scrape the time series from the local Prometheus instance. If you use this method, you only need to open the Prometheus API port, which is 9090.

For more information on Prometheus federation, see https://prometheus.io/docs/prometheus/latest/federation/.

You can also proxy requests through the network boundary. Tools like PushProx deploy a proxy and a client on both sides of the network barrier and allow Prometheus to work across network topologies such as NAT.

For more information on PushProx, see https://github.com/RobustPerception/PushProx.

Reverse Proxy Setup

Prometheus fetches metrics using a pull mechanism, so the server must be able to establish TCP connections to each exporter on the monitored clients. To simplify your firewall configuration, you can use reverse proxy for your exporters to expose all metrics on a single port.

Procedure: Installing Prometheus Exporters with Reverse Proxy

1. In the Uyuni {webui}, open the details page of the system to be monitored, and navigate to the `Formulas` tab.

2. Check the `Prometheus Exporters` checkbox to enable the exporters formula, and click btn:[Save].

3. Navigate to the `Prometheus Exporters` tab in the top menu.

4. Check the `Enable reverse proxy` option, and enter a valid reverse proxy port number. For example, `9999`.

5. Customize the other exporters according to your needs.

6. Click btn:[Save Formula].

7. Apply the highstate and confirm that it completes successfully.

For more information about the monitoring formulas, see **[Salt › Formula-monitoring ›]**.

Security

Prometheus server and Prometheus node exporter offer a built-in mechanism to secure their endpoints with TLS encryption and authentication. Uyuni {webui} simplifies the configuration of all involved components. The TLS certificates have to be provided and deployed by the user. Uyuni offers enabling the following security model:

- Node exporter: TLS encryption and client certificate based authentication

- Prometheus: TLS encryption and basic authentication

For more information about configuring all available options, see **[Salt › Formula-monitoring ›]**.

Generating TLS certificates

By default, Uyuni does not provide any certificates for securing monitoring configuration. For providing security, you can generate or import custom certificates, self-signed or signed by a third party certificate authority (CA).

This section demonstrates how to generate client/server certificates for Prometheus and Node exporter minions self-signed with SUSE Manager CA.

Procedure: Creating server/client TLS certificate

1. On the Uyuni Server, at the command prompt, run following command:

```
rhn-ssl-tool --gen-server --dir="/root/ssl-build" --set-country="COUNTRY" \
--set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
--set-org-unit="ORGANIZATION UNIT" --set-email="name@example.com" \
--set-hostname="minion.example.com" --set-cname="minion.example.com" --no-rpm
```

Ensure that the `set-cname` parameter is the fully qualified domain name (FQDN) of your Salt client. You can use the the the `set-cname` parameter multiple times if you require multiple aliases.

2. Copy `server.crt` and `server.key` files to the Salt minion and provide read access for `prometheus` user.

Organizations

Organizations are used to manage user access and permissions within Uyuni.

For most environments, a single organization is enough. However, more complicated environments might need several organizations. You might like to have an organization for each physical location within your business, or for different business functions.

When you have created your organizations, you can create and assign users to your organizations. You can then assign permissions on an organization level, which applies by default to every user assigned to the organization.

You can also configure authentication methods for your new organization, including PAM and single sign-on. For more information about authentication, see **[Administration › Auth-methods ›]**.

> **NOTE** You must be logged in as the Uyuni administrator to create and manage organizations.

Procedure: Creating a New Organization

1. In the Uyuni {webui}, navigate to menu:Admin[Organizations], and click btn:[Create Organization].

2. In the `Create Organization` dialog, complete these fields:

   ○ In the `Organization Name` field, type a name for your new organization. The name should be between 3 and 128 characters long.

   ○ In the `Desired Login` field, type the login name you want to use for the organization's administrator. This must be a new administrator account, you are not be able to use an existing administrator account to sign in to the new organization, including the one you are currently signed in with.

- In the `Desired Password` field, type a password for the new organization's administrator. Confirm the password by typing it again in the `Confirm Password` field. Password strength is indicated by the colored bar beneath the password fields.

- In the `Email` field, type an email address for the new organization's administrator.

- In the `First Name` field, select a salutation, and type a given name for the new organization's administrator.

- In the `Last Name` field, type a surname for the new organization's administrator.

3. Click btn:[Create Organization].

Manage Organizations

In the Uyuni {webui}, navigate to menu:Admin[Organizations] to see a list of available organizations. Click the name of an organization to manage it.

From the menu:Admin[Organizations] section, you can access tabs to manage users, trusts, configuration, and states for your organization.

NOTE Organizations can only be managed by their administrators. To manage an organization, ensure you are signed in as the correct administrator for the organization you want to change.

Organization Users

Navigate to the `Users` tab to view the list of all users associated with the organization, and their role. Clicking a username takes you to the `Users` menu to add, change, or delete users.

Trusted Organizations

Navigate to the `Trusts` tab to add or remove trusted organizations. Establishing trust between organizations allow them to share content between them, and gives you the ability to transfer clients from one organization to another.

Configure Organizations

Navigate to the `Configuration` tab to manage the configuration of your organization. This includes the use of staged contents, and the use of SCAP files.

For more information about content staging, see **[Administration › Content-staging ›]**.

For more information about OpenSCAP, see **[Reference › Audit ›]**.

Manage States

Navigate to the `States` tab to manage Salt states for all clients in your organization. States allow you to define global security policies, or add a common admin user to all clients.

For more information about Salt States, see **[Salt › Salt-states ›]**.

Manage Configuration Channels

You can select which configuration channels should be applied across your organization. Configuration channels can be created in the Uyuni {webui} by navigating to menu:Configuration[Channels]. Apply configuration channels to your organization using the Uyuni {webui}.

Procedure: Applying Configuration Channels to an Organization

1. In the Uyuni {webui}, navigate to menu:Home[My Organization > Configuration Channels].

2. Use the search feature to locate a channel by name.

3. Check the channel to be applied and click btn:[Save Changes]. This saves to the database, but does not apply the changes to the channel.

4. Apply the changes by clicking btn:[Apply]. This schedules the task to apply the changes to all clients within the organization.

Content Staging

Staging is used by clients to download packages in advance, before they are installed. This allows package installation to begin as soon as it is scheduled, which can reduce the amount of time required for a maintenance window.

Enable Content Staging

You can manage content staging across your entire organization. In the Uyuni {webui}, navigate to menu:Admin[Organizations] to see a list of available organizations. Click the name of an organization, and check the `Enable Staging Contents` box to allow clients in this organization to stage package data.

NOTE   You must be logged in as the Uyuni administrator to create and manage organizations.

You can also enable staging at the command prompt by editing `/etc/sysconfig/rhn/up2date`, and adding or editing these lines:

```
stagingContent=1
stagingContentWindow=24
```

The `stagingContentWindow` parameter is a time value expressed in hours and determines when

downloading starts. It is the number of hours before the scheduled installation or update time. In this example, content is downloaded 24 hours before the installation time. The start time for download depends on the selected contact method for a system. The assigned contact method sets the time for when the next `mgr_check` is executed.

Next time an action is scheduled, packages are automatically downloaded, but not installed. At the scheduled time, the staged packages are installed.

Configure Content Staging

There are two parameters used to configure content staging:

- `salt_content_staging_advance` is the advance time for the content staging window to open, in hours. This is the number of hours before installation starts, that package downloads can begin.

- `salt_content_staging_window` is the duration of the content staging window, in hours. This is the amount of time clients have to stage packages before installation begins.

For example, if `salt_content_staging_advance` is set to six hours, and `salt_content_staging_window` is set to two hours, the staging window opens six hours before the installation time, and remain open for two hours. No packages are downloaded in the four remaining hours until installation starts.

If you set the same value for both `salt_content_staging_advance` and `salt_content_staging_window` packages are able to be downloaded until installation begins.

Configure the content staging parameters in `/usr/share/rhn/config-defaults/rhn_java.conf`.

Default values:

- `salt_content_staging_advance: 8 hours`
- `salt_content_staging_window: 8 hours`

> **NOTE** Content staging must be enabled for these parameters to work correctly.

Disconnected Setup

When it is not possible to connect Uyuni to the internet, you can use it within a disconnected environment.

The repository mirroring tool (RMT) is available on {sle} 15 and later. RMT replaces the subscription management tool (SMT), which can be used on older {sle} installations.

In a disconnected Uyuni setup, RMT or SMT uses an external network to connect to {scc}. All software channels and repositories are synchronized to a removable storage device. The storage device can then be

used to update the disconnected Uyuni installation.

This setup allows your Uyuni installation to remain in an offline, disconnected environment.

> **NOTE** Your RMT or SMT instance must be used to manage the Uyuni Server directly. It cannot be used to manage a second RMT or SMT instance, in a cascade.

For more information on RMT, see https://documentation.suse.com/sles/15-SP3/html/SLES-all/book-rmt.html.

Synchronize RMT

You can use RMT on {sle} 15 installations to manage clients running {sle} 12 or later.

We recommend you set up a dedicated RMT instance for each Uyuni installation.

Procedure: Setting up RMT

1. On the RMT instance, install the RMT package:

```
zypper in rmt-server
```

2. Configure RMT using {yast}:

```
yast2 rmt
```

3. Follow the prompts to complete installation.

For more information on setting up RMT, see https://documentation.suse.com/sles/15-SP3/html/SLES-all/book-rmt.html.

Procedure: Synchronizing RMT with SCC

1. On the RMT instance, list all available products and repositories for your organization:

```
rmt-cli products list --all
rmt-cli repos list --all
```

2. Synchronize all available updates for your organization:

```
rmt-cli sync
```

   You can also configure RMT to synchronize regularly using systemd.

3. Enable the products you require. For example, to enable SLES 15:

```
rmt-cli product enable sles/15/x86_64
```

4. Export the synchronized data to your removable storage. In this example, the storage medium is mounted at `/mnt/usb`:

```
rmt-cli export data /mnt/usb
```

5. Export the enabled repositories to your removable storage:

```
rmt-cli export settings /mnt/usb
rmt-cli export repos /mnt/usb
```

IMPORTANT   Ensure that the external storage is mounted to a directory that is writeable by the RMT user. You can change RMT user settings in the `cli` section of `/etc/rmt.conf`.

Synchronize SMT

SMT is included with {sle} 12, and can be used to manage clients running {sle} 10 or later.

SMT requires you to create a local mirror directory on the SMT instance to synchronize repositories and packages.

For more details on installing and configuring SMT, see https://documentation.suse.com/sles/12-SP5/html/SLES-all/book-smt.html.

Procedure: Synchronizing SMT with SCC

1. On the SMT instance, create a database replacement file:

```
smt-sync --createdbreplacementfile /tmp/dbrepl.xml
```

2. Export the synchronized data to your removable storage. In this example, the storage medium is mounted at `/mnt/usb`:

```
smt-sync --todir /mnt/usb
smt-mirror --dbreplfile /tmp/dbrepl.xml --directory /mnt/usb \
          --fromlocalsmt -L /var/log/smt/smt-mirror-export.log
```

IMPORTANT   Ensure that the external storage is mounted to a directory that is writeable by the RMT user. You can change SMT user settings in `/etc/smt.conf`.

Synchronize a Disconnected Server

When you have removable media loaded with your {scc} data, you can use it to synchronize your disconnected server.

Procedure: Synchronizing a Disconnected Server

1. Mount your removable media device to the Uyuni server. In this example, the mount point is `/media/disk`.

2. Open `/etc/rhn/rhn.conf` and define the mount point by adding or editing this line:

   ```
   server.susemanager.fromdir = /media/disk
   ```

3. Restart the Tomcat service:

   ```
   systemctl restart tomcat
   ```

4. Refresh the local data:

   ```
   mgr-sync refresh
   ```

5. Perform a synchronization:

   ```
   mgr-sync list channels
   mgr-sync add channel channel-label
   ```

> **WARNING**
>
> The removable disk that you use for synchronization must always be available at the same mount point. Do not trigger a synchronization, if the storage medium is not mounted. This results in data corruption.

Content Lifecycle Management

Content lifecycle management allows you to customize and test packages before updating production clients. This is especially useful if you need to apply updates during a limited maintenance window.

Content lifecycle management allows you to select software channels as sources, adjust them as required for your environment, and thoroughly test them before installing onto your production clients.

While you cannot directly modify vendor channels, you can clone them and then modify the clones by adding or removing packages and custom patches. You can assign these cloned channels to test clients to ensure they work as expected. Then, when all tests pass, you can promote them to production servers.

This is achieved through a series of environments that your software channels can move through on their

lifecycle. Most environment lifecycles include at least test and production environments, but you can have as many environments as you require.

This section covers the basic content lifecycle procedures, and the filters available. For more specific examples, see **[Administration › Content-lifecycle-examples ›]**.

Create a Content Lifecycle Project

To set up a content lifecycle, you need to begin with a project. The project defines the software channel sources, the filters used to find packages, and the build environments.

Procedure: Creating a Content Lifecycle Project

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Projects], and click btn:[Create Project].

2. In the `Label` field, enter a label for your project. The `Label` field only accepts lowercase letters, numbers, periods, hyphens, and underscores.

3. In the `Name` field, enter a descriptive name for your project.

4. Click the btn:[Create] button to create your project and return to the project page.

5. Click btn:[Attach/Detach Sources].

6. In the `Sources` dialog, select the source type, and select a base channel for your project. The available child channels for the selected base channel are displayed, including information on whether the channel is mandatory or recommended.

7. Check the child channels you require, and click btn:[Save] to return to the project page. The software channels you selected should now be showing.

8. Click btn:[Attach/Detach Filters].

9. In the `Filters` dialog, select the filters you want to attach to the project. To create a new filter, click btn:[Create new Filter].

10. Click btn:[Add Environment].

11. In the `Environment Lifecycle` dialog, give the first environment a name, a label, and a description, and click btn:[Save]. The `Label` field only accepts lowercase letters, numbers, periods, hyphens, and underscores.

12. Continue creating environments until you have all the environments for your lifecycle completed. You can select the order of the environments in the lifecycle by selecting an environment in the `Insert before` field when you create it.

Filter Types

Uyuni allows you to create various types of filters to control the content used for building the project. Filters allow you to select which packages are included or excluded from the build. For example, you could exclude all kernel packages, or include only specific releases of some packages.

The supported filters are:

- package filtering
  - by name
  - by name, epoch, version, release, and architecture
- patch filtering
  - by advisory name
  - by advisory type
  - by synopsis
  - by keyword
  - by date
  - by affected package
- module
  - by stream

> **NOTE** Package dependencies are not resolved during content filtering.

There are multiple matchers you can use with the filter. Which ones are available depends on the filter type you choose.

The available matchers are:

- contains
- matches (must take the form of a regular expression)
- equals
- greater
- greater or equal
- lower or equal
- lower
- later or equal

Filter `rule` Parameter

Each filter has a `rule` parameter that can be set to either `Allow` or `Deny`. The filters are processed like this:

- If a package or patch satisfies a `Deny` filter, it is excluded from the result.
- If a package or patch satisfies an `Allow` filter, it is included in the result (even if it was excluded by a `Deny` filter).

This behavior is useful when you want to exclude large number of packages or patches using a general `Deny` filter and "cherry-pick" specific packages or patches with specific `Allow` filters.

> **NOTE** Content filters are global in your organization and can be shared between projects.

> **NOTE** If your project already contains built sources, when you add an environment it is automatically populated with the existing content. Content is drawn from the previous environment of the cycle if it had one. If there is no previous environment, it is left empty until the project sources are built again.

Filter Templates

To help with creating filters for some common scenarios, Uyuni offers filter templates. When applied, these templates help creating a set of filters in advance, tailored for a specific use case.

This section describes available templates and their usages.

Live patching based on a SUSE product

In a project that contains live patching, regular kernel packages must be excluded so that only live patch packages are offered as updates to clients. On the other hand, already installed regular kernel packages must still be included to keep system integrity.

When applied, this template creates two filters required to achieve this behavior:

- Deny patches that contain `kernel-default` package greater than a specific kernel version

- Deny patches that contain `reboot_suggested` keyword

> **IMPORTANT** The second "catch-all" filter is a safeguard to make sure the assigned clients never require a restart in the future. You can choose not to attach it to the project if this behavior is not desired as it might block other important updates.

For more information on how to set up a live patching project, see administration:content-lifecycle-examples.pdf.

Procedure: Applying the template

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Filters], and click btn:[Create Filter].

2. In the dialog, click btn:[Use a template]. The inputs will change accordingly.

3. In the `Prefix` field, type a name prefix. This value will be prepended to the name of every individual filter created by the template. If the template is being applied in the context of a project, this field will be prefilled with the project label.

4. In the `Template` field, select `Live patching based on a SUSE product`.

5. In the `Product` field, select the product you wish to set up live patching for.

6. In the `Kernel` field, select a kernel version from the list of versions available in the selected product. The filter to deny the later regular kernel patches will be based on this version.

7. Click btn:[Save] to create the filters.

8. Navigate to menu:Content Lifecycle[Projects] and select your project.

9. Click btn:[Attach/Detach Filters].

10. Select the two filters that have the specified prefix, and click btn:[Save].

Live patching based on a system

When you would like to set up a live patching project based on a kernel version installed in a specific registered system, you might prefer the "live patching based on a system" template.

When applied, this template creates two filters required to achieve this behavior:

- Deny patches that contain `kernel-default` package greater than a specific kernel version

- Deny patches that contain `reboot_suggested` keyword

> **IMPORTANT**
>
> The second "catch-all" filter is a safeguard to make sure the assigned clients never require a restart in the future. You can choose not to attach it to the project if this behavior is not desired as it might block other important updates.

For more information on how to set up a live patching project, see administration:content-lifecycle-examples.pdf.

Procedure: Applying the template

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Filters], and click btn:[Create Filter].

2. In the dialog, click btn:[Use a template]. The inputs will change accordingly.

3. In the `Prefix` field, type a name prefix. This value will be prepended to the name of every individual filter created by the template. If the template is being applied in the context of a project, this field will be prefilled with the project label.

4. In the `Template` field, select `Live patching based on a specific system`.

5. In the `System` field, select a system from the list, or start typing a system name to narrow down the options.

6. In the `Kernel` field, select a kernel version from the list of versions installed in the selected system. The filter to deny the later regular kernel patches will be based on this version.

7. Click btn:[Save] to create the filters.

8. Navigate to menu:Content Lifecycle[Projects] and select your project.

9. Click btn:[Attach/Detach Filters].

10. Select the two filters that have the specified prefix, and click btn:[Save].

Live patching based on a system

When you want to set up a live patching project based on a kernel version installed in a specific registered system, you can use the "live patching based on a system" template.

When applied, this template creates two filters required to achieve this behavior:

- Deny patches that contain `kernel-default` package greater than a specific kernel version

- Deny patches that contain `reboot_suggested` keyword

> **IMPORTANT**
>
> The second "catch-all" filter is a safeguard to make sure the assigned clients never require a restart. You can choose not to attach it to the project if this behavior is not wanted because it might block other important updates.

For more information on how to set up a live patching project, see administration:content-lifecycle-examples.pdf.

Procedure: Applying the template

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Filters], and click btn:[Create Filter].

2. In the dialog, click btn:[Use a template]. The inputs will change accordingly.

3. In the `Prefix` field, type a name prefix. This value will be prepended to the name of every filter created by the template. If the template is being applied in the context of a project, this field will be prefilled with the project label.

4. In the `Template` field, select `Live patching based on a specific system`.

5. In the `System` field, select a system from the list, or start typing a system name to narrow down the options.

6. In the `Kernel` field, select a kernel version from the list of versions installed in the selected system. The filter to deny the later regular kernel patches will be based on this version.

7. Click btn:[Save] to create the filters.

8. Navigate to menu:Content Lifecycle[Projects] and select your project.

9. Click btn:[Attach/Detach Filters].

10. Select the two filters that have the specified prefix, and click btn:[Save].

AppStream modules with defaults

When you want to have all the modules available in a modular repository included in your project, you can automatically add them using this filter template.

When applied, this template creates an AppStream filter per module and its default stream.

If this process is done from the project's page, the filters are added to the project automatically. Otherwise, the created filters can be listed in menu:Content Lifecycle[Filters] and be added to any project as needed.

Each individual filter can be edited to select a different module stream, or removed altogether to exclude that module from the target repositories.

IMPORTANT	Because not all module streams are compatible with each other, changing individual streams may prevent successful resolution of modular dependencies. When this happens, the filters pane in the project details page will show an error describing the problem, and the build button will be disabled until all the module selections are compatible.

For more information on how to set up AppStream repositories with content lifecycle management, see administration:content-lifecycle-examples.pdf.

Procedure: Applying the template

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Projects], and select your project.

2. In the `Filters` section, click btn:[Attach/Detach Filters], and then click btn:[Create New Filter].

3. In the dialog, click btn:[Use a template]. The inputs will change accordingly.

4. In the `Prefix` field, type a name prefix. This value will be prepended to the name of every filter created by the template. If the template is being applied in the context of a project, this field will be prefilled with the project label.

5. In the `Template` field, select `AppStream modules with defaults`.

6. In the `Channel` field, select a modular channel to get the modules from. In this dropdown, only the modular channels are displayed.

7. Click btn:[Save] to create the filters.

8. Scroll to the `Filters` section to see the newly attached AppStream filters.

9. You can edit/remove any individual filter to tailor the project to your needs.

Build a Content Lifecycle Project

When you have created your project, defined environments, and attached sources and filters, you can build the project for the first time.

Building applies filters to the attached sources and clones them to the first environment in the project.

You can use the same vendor channels as sources for multiple content projects. In this case, Uyuni does not create new patch clones for each cloned channel. Instead, a single patch clone is shared between all of your cloned channels. This can cause problems if a vendor modifies a patch; for example, if the patch is retracted, or the packages within the patch are changed. When you build one of the content projects, all

the channels that share the cloned patch are synchronized with the original by default, even if the channels are in other environments of your content project, or other content project channels in your organization. You can change this behavior by turning off automatic patch synchronization in your organization settings. To manually synchronize the patch later for all channels sharing the patch, navigate to menu:Software[Manage > Channels], click the channel you want to synchronize and navigate to the `Sync` subtab. Even manual patch synchronization affects all organization channels sharing the patch.

Procedure: Building a Content Lifecycle Project

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Projects], and select the project you want to build.

2. Review the attached sources and filters, and click btn:[Build].

3. Provide a version message to describe the changes or updates in this build.

4. You can monitor build progress in the `Environment Lifecycle` section.

After the build is finished, the environment version is increased by one and the built sources, such as software channels, can be assigned to your clients.

Promote Environments

When the project has been built, the built sources can be sequentially promoted to the environments.

Procedure: Promoting Environments

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Projects], and select the project you want to work with.

2. In the `Environment Lifecycle` section, locate the environment to promote to its successor, and click btn:[Promote].

3. You can monitor build progress in the `Environment Lifecycle` section.

Assign Clients to Environments

When you build and promote content lifecycle projects, Uyuni creates a tree of software channels. To add clients to the environment, assign the base and child software channels to your client using menu:Software[Software Channels] in the `System Details` page for the client.

NOTE

Newly added cloned channels are not assigned to clients automatically. If you add or promote sources you need to manually check and update your channel assignments.

Automatic assignment is intended to be added to Uyuni in a future version.

Content Lifecycle Management Examples

This section contains some common examples of how you can use content lifecycle management. Use these examples to build your own personalized implementation.

Creating a Project for a Monthly Patch Cycle

An example project for a monthly patch cycle consists of:

- Creating a `By Date` filter

- Adding the filter to the project

- Applying the filter to a new project build

- Excluding a patch from the project

- Including a patch in the project

Creating a `By Date` filter

The `By Date` filter excludes all patches released after a specified date. This filter is useful for your content lifecycle projects that follow a monthly patch cycle.

Procedure: Creating the `By Date` *Filter*

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Filters] and click btn:[Create Filter].

2. In the `Filter Name` field, type a name for your filter. For example, `Exclude patches by date`.

3. In the `Filter Type` field, select `Patch (Issue date)`.

4. In the `Matcher` field, `later or equal` is autoselected.

5. Select the date and time.

6. Click btn:[Save].

Adding the Filter to the Project

Procedure: Adding a Filter to a Project

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Projects] and select a project from the list.

2. Click btn:[Attach/Detach Filters] link to see all available filters

3. Select the new `Exclude patches by date` filter.

4. Click btn:[Save].

Applying the Filter to a new Project Build

The new filter is added to your filter list, but it still needs to be applied to the project. To apply the filter you need to build the first environment.

Procedure: Using the Filter

1. Click btn:[Build] to build the first environment.

2. OPTIONAL: Add a message. You can use messages to help track the build history.

3. Check that the filter has worked correctly by using the new channels on a test server.

4. Click btn:[Promote] to move the content to the next environment. The build takes longer if you have a large number of filters, or they are very complex.

Excluding a Patch from the Project

Tests may help you discover issues. When an issue is found, exclude the problem patch released before the `by date` filter.

Procedure: Excluding a Patch

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Filters] and click btn:[Create Filter].

2. In the `Filter Name` field, enter a name for the filter. For example, `Exclude openjdk patch`.

3. In the `Filter Type` field, select `Patch (Advisory Name)`.

4. In the `Matcher` field, select `equals`.

5. In the `Advisory Name` field, type a name for the advisory. For example, `SUSE-15-2019-1807`.

6. Click btn:[Save].

7. Navigate to menu:Content Lifecycle[Projects] and select your project.

8. Click btn:[Attach/Detach Filters] link, select `Exclude openjdk patch`, and click btn:[Save].

When you rebuild the project with the btn:[Build] button, the new filter is used together with the `by date` filter we added before.

Including a Patch in the Project

In this example, you have received a security alert. An important security patch was released several days after the first of the month you are currently working on. The name of the new patch is `SUSE-15-2019-2071`. You need to include this new patch into your environment.

> **IMPORTANT**
>
> The `Allow` filters rule overrides the exclude function of the `Deny` filter rule. For more information, see **[Administration › Content-lifecycle ›]**.

Procedure: Including a Patch in a Project

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Filters] and click btn:[Create Filter].

2. In the `Filter Name` field, type a name for the filter. For example, `Include kernel security fix`.

3. In the `Filter Type` field, select `Patch (Advisory Name)`.

4. In the `Matcher` field, select `equals`.

5. In the `Advisory Name` field, type `SUSE-15-2019-2071`, and check `Allow`.

6. Click btn:[Save] to store the filter.

7. Navigate to menu:Content Lifecycle[Projects] and select your project from the list.

8. Click btn:[Attach/Detach Filters], and select `Include kernel security patch`.

9. Click btn:[Save].

10. Click btn:[Build] to rebuild the environment.

Update an Existing Monthly Patch Cycle

When a monthly patch cycle is complete, you can update the patch cycle for the next month.

Procedure: Updating a Monthly Patch Cycle

1. In the `by date` field, change the date of the filter to the next month. Alternatively, create a new filter and change the assignment to the project.

2. Check if the exclude filter for `SUSE-15-2019-1807` can be detached from the project. There may be a new patch available to fix this issue.

3. Detach the `allow` filter you added previously. The patch is included by default.

4. Rebuild the project to create a new environment with patches for the next month.

Enhance a Project with Live Patching

This section covers setting up filters to create environments for live patching.

IMPORTANT

When you are preparing to use live patching, there are some important considerations

- Only ever use one kernel version on your systems. The live patching packages are installed with a specific kernel.

- Live patching updates are shipped as one patch.

- Each kernel patch that begins a new series of live patching kernels displays the `required reboot` flag. These kernel patches come with live patching tools. When you have installed them, you need to reboot the system at least once before the next year.

- Only install live patch updates that match the installed kernel version.

- Live patches are provided as stand-alone patches. You must exclude all regular kernel patches with higher kernel version than the currently installed one.

Exclude Packages with a Higher Kernel Version

In this example you update your systems with the `SUSE-15-2019-1244` patch. This patch contains `kernel-default-4.12.14-150.17.1-x86_64`.

You need to exclude all patches which contain a higher version of `kernel-default`.

Procedure: Excluding Packages with a Higher Kernel Version

1. In the Uyuni {webui}, navigate to menu:Content Lifecycle[Filters], and click btn:[Create Filter].

2. In the `Filter Name` field, type a name for your filter. For example, `Exclude kernel greater than 4.12.14-150.17.1`.

3. In the `Filter Type` field, select `Patch (Contains Package)`.

4. In the `Matcher` field, select `version greater than`.

5. In the `Package Name` field, type `kernel-default`.

6. Leave the the `Epoch` field empty.

7. In the `Version` field, type `4.12.14`.

8. In the `Release` field, type `150.17.1`.

9. Click btn:[Save] to store the filter.

10. Navigate to menu:Content Lifecycle[Projects] and select your project.

11. Click btn:[Attach/Detach Filters].

12. Select `Exclude kernel greater than 4.12.14-150.17.1`, and click btn:[Save].

When you click btn:[Build], a new environment is created. The new environment contains all the kernel patches up to the version you installed.

NOTE All kernel patches with higher kernel versions are removed. Live patching kernels remain available as long as they are not the first in a series.

NOTE This procedure can be automated using a filter template. For more information on how to apply a live patching filter template, see administration:content-lifecycle.pdf.

Switch to a New Kernel Version for Live Patching

Live Patching for a specific kernel version is only available for one year. After one year you must update the kernel on your systems. Execute these environment changes:

Procedure: Switch to a New Kernel Version

1. Decide which kernel version to upgrade to. For example: `4.12.14-150.32.1`

2. Create a new kernel version Filter.

3. Detach the previous filter **Exclude kernel greater than 4.12.14-150.17.1** and attach the new filter.

Click btn:[Build] to rebuild the environment. The new environment contains all kernel patches up to the new kernel version you selected. Systems using these channels have the kernel update available for installation. You need to reboot systems after they have performed the upgrade. The new kernel remains

valid for one year. All packages installed during the year match the current live patching kernel filter.

AppStream Filters

If you are using {rhel} 8 clients, you cannot perform package operations such as installing or upgrading directly from modular repositories like the {rhel} AppStream repository. You can use the AppStream filter to transform modular repositories into regular repositories. It does this by keeping the packages in the repository and stripping away the module metadata. The resulting repository can be used in Uyuni in the same way as a regular repository.

The AppStream filter selects a single module stream to be included in the target repository. You can add multiple filters to select multiple module streams.

If you do not use an AppStream filter in your CLM project, the module metadata in the modular sources remains intact, and the target repositories contain the same module metadata. As long as at least one AppStream filter is enabled in the CLM project, all target repositories are transformed into regular repositories.

To use the AppStream filter, you need a CLM project with a modular repository such as `{rhel} AppStream`. Ensure that you included the module you need as a source before you begin.

Procedure: Using AppStream Filters

1. In the Uyuni {webui}, navigate to your {rhel} 8 CLM project. Ensure that you have included the AppStream channels for your project.

2. Click btn:`Create Filter` and use these parameters:

   ° In the `Filter Name` field, type a name for the new filter.

   ° In the `Filter Type` field, select `Module (Stream)`.

   ° In the `Module Name` field, type a module name. For example, `postgresql`.

   ° In the `Stream` field, type the name of the desired stream. For example, `10`. If you leave this field blank, the default stream for the module is selected.

3. Click btn:[Save] to create the new filter.

4. Navigate to menu:Content Lifecycle[Projects] and select your project.

5. Click btn:`Attach/Detach Filters`, select your new AppStream filter, and click btn:[Save].

You can use the browse function in the `Create/Edit Filter` form to select a module from a list of available module streams for a modular channel.

Procedure: Browsing Available Module Streams

1. In the Uyuni {webui}, navigate to your {rhel} 8 CLM project. Ensure that you have included the AppStream channels for your project.

2. Click btn:`Create Filter` and use these parameters:

- ◦ In the `Filter Name` field, type a name for the new filter.

- ◦ In the `Filter Type` field, select `Module (Stream)`.

3. Click `Browse available modules` to see all modular channels.

4. Select a channel to browse the modules and streams:

   - ◦ In the `Module Name` field, start typing a module name to search, or select from the list.

   - ◦ In the `Stream` field, start typing a stream name to search, or select from the list.

NOTE	Channel selection is only for browsing modules. The selected channel is not be saved with the filter, and does not affect the CLM process in any way.

You can create additional AppStream filters for any other module stream to be included in the target repository. Any module streams that the selected stream depends on is automatically included.

NOTE	Be careful not to specify conflicting, incompatible, or missing module streams. For example, selecting two streams from the same module is invalid.

When you build your CLM project using the btn:[Build] button in the {webui}, the target repository is a regular repository without any modules, that contains packages from the selected module streams.

Authentication Methods

Uyuni supports several different authentication methods. This section discusses pluggable authentication modules (PAM) and single sign-on (SSO).

Authentication With Single Sign-On (SSO)

Uyuni supports single sign-on (SSO) by implementing the Security Assertion Markup Language (SAML) 2 protocol.

Single sign-on is an authentication process that allows a user to access multiple applications with one set of credentials. SAML is an XML-based standard for exchanging authentication and authorization data. A SAML identity service provider (IdP) provides authentication and authorization services to service providers (SP), such as Uyuni. Uyuni exposes three endpoints which must be enabled for single sign-on.

SSO in Uyuni supports:

- Log in with SSO.

- Log out with service provider-initiated single logout (SLO), and Identity service provider single logout service (SLS).

- Assertion and nameId encryption.

- Assertion signatures.

- Message signatures with AuthNRequest, LogoutRequest, and LogoutResponses.

- Enable an Assertion consumer service endpoint.

- Enable a single logout service endpoint.

- Publish the SP metadata (which can be signed).

SSO in Uyuni does not support:

- Product choosing and implementation for the identity service provider (IdP).

- SAML support for other products (check with the respective product documentation).

For an example implementation of SSO, see **[Administration › Auth-methods-sso-example ›]**.

IMPORTANT	If you change from the default authentication method to single sign-on, the new SSO credentials apply only to the {webui}. Client tools such as `mgr-sync` or `spacecmd` continue to work with the default authentication method only.

Prerequisites

Before you begin, you need to have configured an external identity service provider with these parameters. Check your IdP documentation for instructions.

IMPORTANT	Your IdP must have a SAML:Attribute containing the username of the IdP user domain, called `uid`. The `uid` attribute passed in the SAML:Attribute must be created in the Uyuni user base before you activate single sign-on.

You need these endpoints:

- Assertion consumer service (or ACS): an endpoint to accept SAML messages to establish a session into the Service Provider. The endpoint for ACS in Uyuni is: https://server.example.com/rhn/manager/sso/acs

- Single logout service (or SLS): an endpoint to initiate a logout request from the IdP. The endpoint for SLS in Uyuni is: https://server.example.com/rhn/manager/sso/sls

- Metadata: an endpoint to retrieve Uyuni metadata for SAML. The endpoint for metadata in Uyuni is: https://server.example.com/rhn/manager/sso/metadata

After the authentication with the IdP using the user `orgadmin` is successful, you are logged in to Uyuni as the `orgadmin` user, provided that the `orgadmin` user exists in Uyuni.

Enable SSO

NOTE Using SSO is mutually exclusive with other types of authentication: it is either enabled or disabled. SSO is disabled by default.

Procedure: Enabling SSO

1. If your users do not yet exist in Uyuni, create them first.

2. Edit `/etc/rhn/rhn.conf` and add this line at the end of the file:

   ```
   java.sso = true
   ```

3. Find the parameters you want to customize in `/usr/share/rhn/config-defaults/rhn_java_sso.conf`. Insert the parameters you want to customize into `/etc/rhn/rhn.conf` and prefix them with `java.sso`. For example, in `/usr/share/rhn/config-defaults/rhn_java_sso.conf` find:

   ```
   onelogin.saml2.sp.assertion_consumer_service.url = https://YOUR-PRODUCT-HOSTNAME-OR-
   IP/rhn/manager/sso/acs
   ```

   To customize it, create the corresponding option in `/etc/rhn/rhn.conf` by prefixing the option name with `java.sso.`:

   ```
   java.sso.onelogin.saml2.sp.assertion_consumer_service.url = https://YOUR-PRODUCT-
   HOSTNAME-OR-IP/rhn/manager/sso/acs
   ```

   To find all the occurrences you need to change, search in the file for the placeholders `YOUR-PRODUCT` and `YOUR-IDP-ENTITY`. Every parameter comes with a brief explanation of what it is meant for.

4. Restart the spacewalk service to pick up the changes:

   ```
   spacewalk-service restart
   ```

When you visit the Uyuni URL, you are redirected to the IdP for SSO where you are requested to authenticate. Upon successful authentication, you are redirected to the Uyuni {webui}, logged in as the authenticated user. If you encounter problems with logging in using SSO, check the Uyuni logs for more information.

Authentication With PAM

Uyuni supports network-based authentication systems using pluggable authentication modules (PAM). PAM is a suite of libraries that allows you to integrate Uyuni with a centralized authentication mechanism, eliminating the need to remember multiple passwords. Uyuni supports LDAP, Kerberos, and other network-based authentication systems using PAM.

Procedure: Enabling PAM

1. Create a PAM service file at `/etc/pam.d/susemanager`. Filename must be in lower case and readable by `tomcat` user. This file is used by Uyuni to load the correct PAM configuration files:

```
#%PAM-1.0
auth     include        common-auth
account  include        common-account
password include        common-password
session  include        common-session
```

On the Uyuni Server, at the command prompt, as root, add the `SSS` *PAM module:*

```
pam-config -a --sss
```

This command adds the module to the `/etc/pam.d/common-auth` configuration file. We do not recommend editing this file directly.

2. Enforce the use of the service file by adding this line to `/etc/rhn/rhn.conf`:

```
pam_auth_service = susemanager
```

```
  In this example, the PAM service file is called [systemitem]``susemanager``.
. Restart the {productname} services after a configuration change.
. In the {productname} {webui}, navigate to menu:Users[Create User] and enable a new or
existing user to authenticate with PAM.
. Check the [guimenu]``Pluggable Authentication Modules (PAM)`` checkbox.
  It is below the password and password confirmation fields.
```

IMPORTANT
Changing the password in the Uyuni {webui} changes only the local password on the Uyuni Server. If PAM is enabled for that user, the local password might not be used at all. In the above example, for instance, the Kerberos password is not changed. Use the password change mechanism of your network service to change the password for these users.

To configure system-wide authentication you can use YaST. You need to install the `yast2-auth-client` package.

For more information about configuring PAM, the SUSE Linux Enterprise Server Security Guide contains a generic example that also works for other network-based authentication methods. It also describes how to configure an active directory service. For more information, see https://documentation.suse.com/sles/15-SP3/html/SLES-all/part-auth.html.

SSL Certificates

Uyuni uses SSL certificates to ensure that clients are registered to the correct server.

Every client that uses SSL to register to the Uyuni Server checks that it is connecting to the right server by validating against a server certificate. This process is called an SSL handshake.

During the SSL handshake, the client checks that the hostname in the server certificate matches what it expects. The client also needs to check if the server certificate is trusted.

Every Uyuni Server that uses SSL requires an SSL server certificate. Provide the path to the server certificate using the `SERVER_CERT` environment variable during setup, or with the `--from-server-cert` option of the `rhn-ssl-tool` command.

Certificate authorities (CAs) are certificates that are used to sign other certificates. All certificates must be signed by a certificate authority (CA) in order for them to be considered valid, and for clients to be able to successfully match against them.

When an organization signs its own certificate, the certificate is considered self-signed. A self-signed certificate is straight-forward to set up, and does not cost any money, but they are considered less secure. If you are using a self-signed certificate, you have a root CA that is signed with itself. When you look at the details of a root CA, you can see that the subject has the same value as the issuer. Provide the path to your root CA certificate using the `CA_CERT` environment variable during setup, or with the `--ca-cert` option of the `rhn-ssl-tool` command.

In order for SSL authentication to work correctly, the client must trust the root CA. This means that the root CA must be installed on every client.

The default method of SSL authentication is for Uyuni to use self-signed certificates. In this case, Uyuni has generated all the certificates, and the root CA has signed the server certificate directly.

An alternative method is to use an intermediate CA. In this case, the root CA signs the intermediate CA. The intermediate CA can then sign any number of other intermediate CAs, and the final one signs the server certificate. This is referred to as a chained certificate.

If you are using intermediate CAs in a chained certificate, the root CA is installed on the client, and the server certificate is installed on the server. During the SSL handshake, clients must be able to verify the entire chain of intermediate certificates between the root CA and the server certificate, so they must be able to access all the intermediate certificates.

There are two main ways of achieving this. In Uyuni, by default, all the intermediate CAs are installed on the client. However, you could also configure your services on the server to provide them to the client. In this case, during the SSL handshake, the server presents the server certificate as well as all the intermediate CAs.

Whichever method you choose, you must ensure that the `CA_CERT` environment variable points to the root CA, and all intermediate CAs. It should not contain the server certificate. The server certificate must be defined at the `SERVER_CERT` environment variable.

By default, Uyuni uses a self-signed certificate. For additional security, you can arrange a third party CA to sign your certificates. Third party CAs perform checks to ensure that the information contained in the certificate is correct. They usually charge an annual fee for this service. Using a third party CA makes

certificates harder to spoof, and provides additional protection for your installation. If you have certificates signed by a third party CA, you can import them to your Uyuni installation.

- For more on self-signed certificates, see **[Administration › Ssl-certs-selfsigned ›]**.

- For more on imported certificates, see **[Administration › Ssl-certs-imported ›]**.

Self-Signed SSL Certificates

By default, Uyuni uses a self-signed certificate. In this case, the certificate is created and signed by Uyuni. This method does not use an independent certificate authority to guarantee that the details of the certificate are correct. Third party CAs perform checks to ensure that the information contained in the certificate is correct. For more on third party CAs, see **[Administration › Ssl-certs-imported ›]**.

This section covers how to re-create your self-signed certificates on an existing installation. It also covers how to create new self-signed certificates and authenticate your existing clients to the new certificate, using an intermediate certificate. Intermediate certificates merge the intermediate and root CA certificates into one file. Ensure that the intermediate certificate comes first in the combined file.

The host name of the SSL keys and certificates must match the fully qualified host name of the machine you deploy them on.

Re-Create Existing Server Certificates

If your existing certificates have expired or stopped working for any reason, you can generate a new server certificate from the existing CA.

Procedure: Re-Creating an Existing Server Certificate

1. On the Uyuni Server, at the command prompt, regenerate the server certificate:

   ```
   rhn-ssl-tool --gen-server --dir="/root/ssl-build" --set-country="COUNTRY" \
   --set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
   --set-org-unit="ORGANIZATION UNIT" --set-email="name@example.com" \
   --set-hostname="susemanager.example.com" --set-cname="example.com"
   ```

   Ensure that the `set-cname` parameter is the fully qualified domain name of your Uyuni Server. You can use the the `set-cname` parameter multiple times if you require multiple aliases.

2. Install the RPM that contains the newly generated certificate. Check that you have the latest version of the RPM before running this command. The version number is incremented every time you re-create the certificates.

   ```
   rpm -Uhv /root/ssl-build/lnx0259a/rhn-org-httpd-ssl-key-pair-lnx0259a-1.0-2.noarch.rpm
   ```

3. Restart services to pick up the changes:

```
spacewalk-service restart
```

Create and Replace CA and Server Certificates

If you need to create entirely new certificates for an existing installation, you need to create a combined certificate first. Clients authenticate to the certificate with both the old and new details. Then you can go ahead and remove the old details. This maintains the chain of trust.

IMPORTANT	Be careful with this procedure. It is possible to break the trust chain between the server and clients using this procedure. If that happens, you need an administrative user to log in to every client and deploy the CA directly.

Procedure: Creating New Certificates

1. On the Uyuni Server, at the command prompt, move the old certificate directory to a new location:

   ```
   mv /root/ssl-build /root/old-ssl-build
   ```

2. Generate a new CA certificate and create an RPM:

   ```
   rhn-ssl-tool --gen-ca --dir="/root/ssl-build" --set-country="COUNTRY" \
   --set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
   --set-org-unit="ORGANIZATION UNIT" --set-common-name="SUSE Manager CA Certificate" \
   --set-email="name@example.com"
   ```

3. Generate a new server certificate and create an RPM:

   ```
   rhn-ssl-tool --gen-server --dir="/root/ssl-build" --set-country="COUNTRY" \
   --set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
   --set-org-unit="ORGANIZATION UNIT" --set-email="name@example.com" \
   --set-hostname="susemanager.example.top" --set-cname="example.com"
   ```

   Ensure that the `set-cname` parameter is the fully qualified domain name of your Uyuni Server. You can use the the `set-cname` parameter multiple times if you require multiple aliases.

   You need to generate a server certificate RPM for each proxy, using their host names and cnames.

When you have new certificates, you can create the combined RPMs to authenticate the clients.

Procedure: Create Combined Certificate RPMs

1. Create a new CA file that combines the old and new certificate details, and generate a new RPM:

```
mkdir /root/combined-ssl-build
cp /root/old-ssl-build/RHN-ORG-TRUSTED-SSL-CERT /root/combined-ssl-build/
cat /root/ssl-build/RHN-ORG-TRUSTED-SSL-CERT >> /root/combined-ssl-build/RHN-ORG-
TRUSTED-SSL-CERT
cp /root/old-ssl-build/*.rpm /root/combined-ssl-build/
rhn-ssl-tool --gen-ca --rpm-only --dir="/root/combined-ssl-build"
```

2. Deploy the CA certificate on the server:

```
/usr/bin/rhn-deploy-ca-cert.pl --source-dir /root/combined-ssl-build \
--target-dir /srv/www/htdocs/pub/ --trust-dir=/etc/pki/trust/anchors/
```

When you have the combined RPMs, you can deploy the combined CA certificates to your clients.

Procedure: Deploying Combined Certificates on Traditional Clients

1. On the Uyuni Server, create a new custom channel using these details:

   ◦ Name: SSL-CA-Channel

   ◦ Label: ssl-ca-channel

   ◦ Parent Channel: <choose the parent channel of a client>

   ◦ Summary: SSL-CA-Channel

   For more on creating custom channels, see **[Administration › Channel-management ›]**.

2. Upload the CA certificate RPM to the channel:

```
rhnpush -c ssl-ca-channel --nosig \
--ca-chain=/srv/www/htdocs/pub/RHN-ORG-TRUSTED-SSL-CERT \
/root/combined-ssl-build/rhn-org-trusted-ssl-cert-1.0-2.noarch.rpm
```

3. Subscribe all clients to the new `SSL-CA-Channel` channel.

4. Install the CA certificate RPM on all clients by updating the channel.

Procedure: Deploying Combined Certificates on Salt Clients

1. In the Uyuni {webui}, navigate to menu:Systems[Overview].

2. Check all your Salt Clients to add them to the system set manager.

3. Navigate to menu:Systems[System Set Manager > Overview].

4. In the `States` field, click btn:[Apply] to apply the system states.

5. In the `Highstate` page, click btn:[Apply Highstate] to propagate the changes to the clients.

When you have every client trusting both the old and new certificates, you can go ahead and replace the server certificate on the Uyuni Server and Proxies.

Procedure: Replace Server Certificate on the Server

1. On the Uyuni Server, at the command prompt, install the RPM from the `ssl-build` directory:

   ```
   rpm -Uhv ssl-build/susemanager/rhn-org-httpd-ssl-key-pair-susemanager-1.0-2.noarch.rpm
   ```

2. Restart services to pick up the changes:

   ```
   spacewalk-service restart
   ```

Procedure: Replace Server Certificate on the Proxy

1. On the Uyuni Proxy, at the command prompt, install the RPM from the `ssl-build` directory:

   ```
   rpm -Uhv ssl-build/susemanager-proxy/rhn-org-httpd-ssl-key-pair-susemanager-proxy-1.0-
   2.noarch.rpm
   ```

2. Restart services to pick up the changes:

   ```
   rhn-proxy restart
   ```

3. Test that all clients still operate as expected and can use SSL to reach the Uyuni Server and any proxies.

When you have replaced the server certificates on your server and any proxies, you need to update the certificate with only the new details on all the clients. This is done by adding it to the client channels you set up previously.

Procedure: Adding the New Certificates to the Client Channel

1. Copy the combined certificate RPM into the `/root/ssl-build/` directory:

   ```
   cp /root/combined-ssl-build/*.rpm /root/ssl-build/
   ```

2. Generate a new RPM with from the new certificates. Check the release number carefully to ensure you have the right certificate file:

   ```
   rhn-ssl-tool --gen-ca --rpm-only --dir="/root/ssl-build"
   ```

3. Install the new local certificates on the Uyuni Server:

   ```
   /usr/bin/rhn-deploy-ca-cert.pl --source-dir /root/ssl-build \
   --target-dir /srv/www/htdocs/pub/ --trust-dir=/etc/pki/trust/anchors/
   ```

4. Restart services to pick up the changes:

```
spacewalk-service restart
```

5. Upload the new RPM into the channel:

```
rhnpush -c ssl-ca-channel --nosig \
--ca-chain=/srv/www/htdocs/pub/RHN-ORG-TRUSTED-SSL-CERT \
/root/ssl-build/rhn-org-trusted-ssl-cert-1.0-3.noarch.rpm
```

When you have the new certificate in the channel, you can use the Uyuni {webui} to update it on all clients and proxies, by synchronizing them with the channel. Alternatively, for Salt clients, you can use menu:Salt[Remote Commands], or apply the highstate.

You also need to update your proxies to remove the copy of the certificate and the associated RPM. Your proxies must have the same certificate content as the server. Check the `/srv/www/htdocs/pub/` directory and ensure it contains:

```
RHN-ORG-TRUSTED-SSL-CERT
rhn-org-trusted-ssl-cert-*.noarch.rpm
```

To complete the process, you need to update the database with this command:

```
/usr/bin/rhn-ssl-dbstore --ca-cert=/root/ssl-build/RHN-ORG-TRUSTED-SSL-CERT
```

If you use bootstrap, remember to also update your bootstrap scripts to reflect the new certificate information.

Import SSL Certificates

By default, Uyuni uses a self-signed certificate. For additional security, you can import a custom certificate, signed by a third party certificate authority (CA).

This section covers how to use an imported SSL certificate with a new Uyuni installation, and how to replace existing self-signed certificates with imported certificates.

Before you begin, ensure you have:

- A certificate authority (CA) SSL public certificate. If you are chaining CAs, it must include the root CA, and all intermediate CAs.
- An SSL server key
- An SSL server certificate

Your key and certificate files must be in PEM format.

The host name of the SSL keys and certificates must match the fully qualified host name of the machine you deploy them on. You can set the host names in the `X509v3 Subject Alternative Name` section of the certificate. You can also list multiple host names if your environment requires it.

Third-party authorities commonly use intermediate CAs to sign requested server certificates. In this case, all CAs in the chain are required to be available in a single file to deploy the CA to the clients.

Import Certificates for New Installations

By default, Uyuni uses a self-signed certificate. After you have completed the initial setup, you can replace the default certificate with an imported certificate.

Procedure: Import Certificates on a New Uyuni Server

1. Install the Uyuni Server according to the instructions in **[Installation › Install-intro ›]**.

2. Complete the initial setup according to **[Installation › Server-setup ›]**.

3. At the command prompt, point the SSL environment variables to the certificate file locations:

```
export CA_CERT=<path_to_CA_certificates_file>
export SERVER_KEY=<path_to_web_server_key>
export SERVER_CERT=<path_to_web_server_certificate>
```

4. Complete Uyuni setup:

```
yast susemanager_setup
```

When you are prompted for certificate details during setup, fill in random values. The values are overridden by the values you specified at the command prompt.

NOTE   Execute the `yast susemanager_setup` command from the same shell you exported the environment variables from.

Import Certificates for New Proxy Installations

By default, Uyuni Proxy uses a self-signed certificate. After you have completed the initial setup, you can replace the default certificate with an imported certificate.

Procedure: Import Certificates on a New Uyuni Proxy

1. Install the Uyuni Proxy according to the instructions in **[Installation › Install-intro ›]**.

2. Complete the initial setup according to **[Installation › Proxy-setup ›]**.

3. At the command prompt, run:

```
configure-proxy.sh
```

4. At the Do you want to import existing certificates? prompt, type kbd:[y].

5. Follow the prompts to complete setup.

> **NOTE** Use the same certificate authority to sign all server certificates for servers and proxies. Certificates signed with different CAs do not match.

Replace Certificates with a Third Party Certificate

You can replace active certificates on your Uyuni installation with a new third party certificate. To replace the certificates, you can replace the installed CA certificate RPM with a new RPM containing the third party certificate, and then update the database.

This procedure is similar to the one described in administration:ssl-certs-selfsigned.pdf. The difference is that we import the certificates generated by an external PKI.

Procedure: Replacing Existing Certificates

1. On the Uyuni Server, at the command prompt, move the old certificate directory to a backup location:

```
mv /root/ssl-build /root/old-ssl-build
```

2. Generate a CA certificate RPM from the new certificate:

```
rhn-ssl-tool --gen-ca --rpm-only --dir="/root/ssl-build" --from-ca
-cert=<Path_to_CA_Certificates>
```

3. Generate a new server certificate RPM:

```
rhn-ssl-tool --gen-server --rpm-only --dir="/root/ssl-build" --from-server
-key=<Server_Key_File> --from-server-cert=<Server_Cert_File>
```

4. Install the RPM that contains the newly generated certificate. Check that you have the latest version of the RPM before running this command. The version number is incremented every time you re-create the certificates.

```
rpm -Uhv /root/ssl-build/sumahostname/rhn-org-httpd-ssl-key-pair-sumahostname-1.0-
2.noarch.rpm
```

5. Restart services to pick up the changes:

```
spacewalk-service restart
```

When you create the new server certificate RPM, you might get a warning that server certificate request file could not be found. This file is not required, and the procedure completes correctly without it. However, if you want to avoid the error, you can copy the file into the server directory, and name it `server.csr`:

```
cp <Certificate_Request_File>.csr /root/ssl-build/<Server_Name>/server.csr
```

When you have created the new `ssl-build` directory, you can create combined certificate RPMs and deploy them on the clients. For the procedures to do this, see **[Administration › Ssl-certs-selfsigned ›]**.

If you are using a proxy, you need to generate a server certificate RPM for each proxy, using their host names and cnames.

Inter-Server Synchronization

If you have more than one Uyuni installation, you need to ensure that they stay aligned on content and permissions. Inter-Server Synchronization (ISS) allows you to connect two or more Uyuni Servers and keep them up-to-date.

To set up ISS, you need to define one Uyuni Server as a master, with the other as a slave. If conflicting configurations exist, the system prioritizes the master configuration.

> **NOTE** ISS Masters are masters only because they have slaves attached to them. This means that you need to set up the ISS Master first, by defining its slaves. You can then set up the ISS Slaves, by attaching them to a master.

Procedure: Setting up an ISS Master

1. In the Uyuni {webui}, navigate to menu:Admin[ISS Configuration > Master Setup], and click btn:[Add new slave].

2. In the `Edit Slave Details` dialog, provide these details for the ISS Master's first slave:

    ◦ In the `Slave Fully Qualified Domain Name` field, enter the FQDN of the ISS Slave. For example: `server2.example.com`.

    ◦ Check the `Allow Slave to Sync?` checkbox to enable the slave to synchronize with the master.

    ◦ Check the `Sync All Orgs to Slave?` checkbox to synchronize all organizations to this slave.

3. Click btn:[Create] to add the ISS slave.

4. In the `Allow Export of the Selected Organizations` section, check the organizations you want to allow this slave to export to the master, and click btn:[Allow Orgs].

Before you set up the ISS Slave, you need to ensure you have the appropriate CA certificate.

Procedure: Copying the Master CA Certificate to an ISS Slave

1. On the ISS Master, locate the CA Certificate at `/srv/www/htdocs/pub/RHN-ORG-TRUSTED-SSL-CERT` and create a copy that can be transferred to the ISS Slave.

2. On the ISS Slave, save the CA certificate file to the `/etc/pki/trust/anchors/` directory.

When you have copied the certificate, you can set up the ISS Slave.

Procedure: Setting up an ISS Slave

1. In the Uyuni {webui}, navigate to menu:Admin[ISS Configuration > Slave Setup], and click btn:[Add new master].

2. In the `Details for new Master` dialog, provide these details for the server to use as the ISS master:

   ◦ In the `Master Fully Qualified Domain Name` field, enter the FQDN of the ISS Master for this slave. For example: `server1.example.com`.

   ◦ In the `Filename of this Master's CA Certificate` field, enter the absolute path to the CA certificate on the ISS master. This should be `/etc/pki/trust/anchors/RHN-ORG-TRUSTED-SSL-CERT`.

3. Click btn:[Add new master] to add the ISS Slave to this master.

Procedure: Completing ISS Setup

1. At the command prompt on the ISS Slave, synchronize with the ISS Master:

   ```
   mgr-inter-sync
   ```

2. OPTIONAL: To synchronize a single channel, use this command:

   ```
   mgr-inter-sync -c <channel-name>
   ```

3. In the Uyuni {webui}, navigate to menu:Admin[ISS Configuration > Configure Master-to-Slave Mappings] and select the organizations you want to synchronize.

Inter-Server Synchronization - Version 2

If you have more than one Uyuni installation, you will need to copy contents between servers. Inter-Server Synchronization (ISS) allows you to export data from one server (source) and import it on another (target) server. This is useful for hub deployment scenarios or disconnected setups.

NOTE   With the version 2 ISS implementation {SUSE} removed the master/slave notion. Contents can be exported and imported in any direction between any Uyuni server.

Install ISS Packages

To use ISS you need to install the `inter-server-sync` package on source and target servers.

Content Synchronization

Procedure: Export data on source server

1. On the source server, on the command line execute the ISS export command. The `-h` option provides detailed help:

```
inter-server-sync export -h
```

The export procedure creates an output directory with all the needed data for the import procedure.

Procedure: Copy export directory to target server

1. Contents from the source server needs to be synchronized to the target server. On the command line, as root, execute:

```
rsync -r <PATH_EXPORTED_DIR> root@<TARGET_SERVER>:~/
```

When all contents is copied, start importing it.

Procedure: Import data on target server

1. On the target server, on the command line execute the ISS import command. The `-h` option provides detailed help:

```
inter-server-sync import -h
```

Database connection configuration

Database connection configuration is loaded by default from `/etc/rhn/rhn.conf`. Properties file location can be overridden with parameter `--serverConfig`.

Known Limitations

- Source and target servers need to be on the same version
- Export and import organization should have the same name

Actions

You can manage actions on your clients in a number of different ways.

For Salt clients, you can schedule automated recurring actions to apply the highstate to clients on a specified schedule. You can apply recurring actions to individual clients, to all clients in a system group, or to an entire organization.

On both Salt and traditional clients, you can set actions to be performed in a particular order by creating action chains. Action chains can be created and edited ahead of time, and scheduled to run at a time that suits you.

You can also perform remote commands on one or more of your Salt clients. Remote commands allows you to issue commands to individual Salt clients, or to all clients that match a search term.

Recurring Actions

You can apply recurring actions on individual Salt clients, or to all clients in an organization.

Procedure: Creating a New Recurring Action

1. To apply a recurring action to an individual client, navigate to `Systems`, click the client to configure schedules for, and navigate to the menu:States[Recurring States] tab.

2. To apply a recurring action to a system group, navigate to menu:Systems[System Groups], select the group to configure schedules for, and navigate to menu:States[Recurring States] tab.

3. Click btn:[Create].

4. Type a name for the new schedule.

5. Choose the frequency of the recurring action:

    ◦ `Hourly:` Type the minute of each hour. For example, `15` runs the action at fifteen minutes past every hour.

    ◦ `Daily:` Select the time of each day. For example, `01:00` runs the action at 0100 every day, in the timezone of the Uyuni Server.

    ◦ `Weekly:` Select the day of the week and the time of the day, to execute the action every week at the specified time.

    ◦ `Monthly:` Select the day of the month and the time of the day, to execute the action every month at the specified time.

    ◦ `Custom Quartz format:` For more detailed options, enter a custom quartz string. For example, to run a recurring action at 0215 every Saturday of every month, enter:

    ```
    0 15 2 ? * 7
    ```

6. OPTIONAL: Toggle the `Test mode` switch on to run the schedule in test mode.

7. Click btn:[Create Schedule] to save, and see the complete list of existing schedules.

Organization Administrators can set and edit recurring actions for all clients in the organization. Navigate to menu:Home[My Organization > Recurring States] to see all recurring actions that apply to the entire

organization.

Uyuni Administrators can set and edit recurring actions for all clients in all organizations. Navigate to menu:Admin[Organizations], select the organization to manage, and navigate to the menu:States[Recurring States] tab.

> **NOTE** Recurring actions can only be used with Salt clients. Traditional clients in your group or organization are ignored.

Action Chains

If you need to perform a number of sequential actions on your clients, you can create an action chain to ensure the order is respected.

By default, most clients execute an action as soon as the command is issued. In some case, actions take a long time, which could mean that actions issued afterwards fail. For example, if you instruct a client to reboot, then issue a second command, the second action could fail because the reboot is still occurring. To ensure that actions occur in the correct order, use action chains.

You can use action chains on both traditional and Salt clients. Action chains can include any number of these actions, in any order:

- menu:System Details[Remote Command]

- menu:System Details[Schedule System Reboot]

- menu:System Details[States > Highstate]

- menu:System Details[Software > Packages > List/Remove]

- menu:System Details[Software > Packages > Install]

- menu:System Details[Software > Packages > Upgrade]

- menu:System Details[Software > Patches]

- menu:System Details[Software > Software Channels]

- menu:System Details[Configuration]

- menu:Images[Build]

Procedure: Creating a New Action Chain

1. In the Uyuni {webui}, navigate to the first action you want to perform in the action chain. For example, navigate to `System Details` for a client, and click btn:[Schedule System Reboot].

2. Check the `Add to` field and select the action chain you want to add to:

   ◦ If this is your first action chain, select `new action chain`.

   ◦ If the action chain already exists, select it from the list.

   ◦ If you already have existing action chains, but you want to create a new action chain, start typing a name for the new action chain to create it.

3. Confirm the action. The action is not performed immediately, it creates the new action chain, and a blue bar confirming this appears at the top of the screen.

4. Continue adding actions to your action chain by checking the `Add to` field and selecting the name of the action chain to add them to.

5. When you have finished adding actions, navigate to menu:Schedule[Action Chains] and selecting the action chain from the list.

6. Re-order actions by dragging them and dropping them into the correct position. Click the blue plus sign to see the clients an action is to be performed on. Click btn:[Save] to save your changes.

7. Schedule a time for your action chain to run, and click btn:[Save and Schedule]. If you leave the page without clicking either btn:[Save] or btn:[Save and Schedule] all unsaved changes are discarded.

> **NOTE**
>
> If one action in an action chain fails, the action chain stops, and no further actions are executed.

You can see scheduled actions from action chains by navigating to menu:Schedule[Pending Actions].

Remote Commands

You can configure clients to run commands remotely. This allows you to issue scripts or individual commands to a client, without having access to the client directly.

This feature is automatically enabled on Salt clients, and you do not need to perform any further configuration. For traditional clients, the feature is enabled if you have registered the client using a bootstrap script and have enabled remote commands. You can use this procedure to enable it manually, instead.

Before you begin, ensure your client is subscribed to the appropriate tools child channel for its installed operating system. For more information about subscribing to software channels, see [**Client-configuration › Channels › **].

Procedure: Configuring Traditional Clients to Accept Remote Commands

1. On the client, at the command prompt, use the package manager to install the `rhncfg`, `rhncfg-client`, and `rhncfg-actions` packages, if not already installed. For example:

   ```
   zypper in rhncfg rhncfg-client rhncfg-actions
   ```

2. On the client, at the command prompt, as root, create a path in the local configuration directory:

   ```
   mkdir -p /etc/sysconfig/rhn/allowed-actions/script
   ```

3. Create an empty file called `run` in the new directory. This file grants the Uyuni Server permission to run remote commands:

```
touch /etc/sysconfig/rhn/allowed-actions/script/run
```

For Salt clients, remote commands are run from the `/tmp/` directory on the client. To ensure that remote commands work accurately, do not mount `/tmp` with the `noexec` option.

> **WARNING**
>
> All commands run from the `Remote Commands` page are executed as {rootuser} on clients. Wildcards can be used to run commands across any number of systems. Always take extra care to check your commands before issuing them.

Procedure: Running Remote Commands on Traditional Clients

1. In the Uyuni {webui}, navigate to `Systems`, click the client to run a remote command on, and navigate to the menu:Details[Remote Command] tab.

2. In the `Run as user` field, type the user ID (UID) of the user on the client that you want to run the command. Alternatively, you can specify a group to run the command, using the group ID (GID) in the `Run as group` field.

3. OPTIONAL: In the `Timeout` field, type a timeout period for the command, in seconds. If the command is not executed within this period, it is not run.

4. In the `Command label` field, type a name for your command.

5. In the `Script` field, type the command or script to execute.

6. Select a date and time to execute the command, or add the remote command to an action chain.

7. Click btn:[Schedule] to schedule the remote command.

For more information about action chains, see **[Reference › Schedule ›]**.

Procedure: Running Remote Commands on Salt Clients

1. Navigate to menu:Salt[Remote Commands].

2. In the first field, before the `@` symbol, type the command you want to issue.

3. In the second field, after the `@` symbol, type the client you want to issue the command on. You can type the `minion-id` of an individual client, or you can use wildcards to target a range of clients.

4. Click btn:[Find targets] to check which clients you have targeted, and confirm that you have used the correct details.

5. Click btn:[Run command] to issue the command to the target clients.

Task Schedules

Under menu:Admin[Task Schedules] all predefined task bunches are listed.

SUSE Manager Schedules ?   + create schedule

Below is a list of defined schedules. A schedule defines frequency, how often a predefined bunch shall be triggered.

1 - 23 of 23

			25 ▾ items per page

Schedule name ↓≡	Frequency	Active From	Bunch
auto-errata-default	0 5/10 * * * ?	2018-06-05 11:40:50 CEST	auto-errata-bunch
channel-repodata-default	0 * * * * ?	2018-06-05 11:40:50 CEST	channel-repodata-bunch
cleanup-data-default	0 0 23 ? * *	2018-06-05 11:40:50 CEST	cleanup-data-bunch
clear-taskologs-default	0 0 23 ? * *	2018-06-05 11:40:50 CEST	clear-taskologs-bunch
cobbler-sync-default	0 * * * * ?	2018-06-05 11:40:50 CEST	cobbler-sync-bunch
compare-configs-default	0 0 23 ? * *	2018-06-05 11:40:50 CEST	compare-configs-bunch
cve-server-channels-default	0 0 23 ? * *	2018-06-05 11:40:51 CEST	cve-server-channels-bunch
daily-status-default	0 0 23 ? * *	2018-06-05 11:40:50 CEST	daily-status-bunch
errata-cache-default	0 * * * * ?	2018-06-05 11:40:50 CEST	errata-cache-bunch
errata-queue-default	0 * * * * ?	2018-06-05 11:40:50 CEST	errata-queue-bunch
gatherer-matcher-default	0 0 0 ? * *	2018-06-05 11:40:51 CEST	gatherer-matcher-bunch
kickstart-cleanup-default	0 0/10 * * * ?	2018-06-05 11:40:50 CEST	kickstart-cleanup-bunch
kickstartfile-sync-default	0 0/10 * * * ?	2018-06-05 11:40:50 CEST	kickstartfile-sync-bunch
mgr-register-default	0 0/15 * * * ?	2018-06-05 11:40:50 CEST	mgr-register-bunch
mgr-sync-refresh-default	0 6 1 ? * *	2018-06-05 11:40:51 CEST	mgr-sync-refresh-bunch
minion-action-cleanup-default	0 0 * * * ?	2018-06-05 11:40:50 CEST	minion-action-cleanup-bunch
package-cleanup-default	0 0/10 * * * ?	2018-06-05 11:40:50 CEST	package-cleanup-bunch
reboot-action-cleanup-default	0 0 * * * ?	2018-06-05 11:40:50 CEST	reboot-action-cleanup-bunch
sandbox-cleanup-default	0 5 4 ? * *	2018-06-05 11:40:50 CEST	sandbox-cleanup-bunch
session-cleanup-default	0 0/15 * * * ?	2018-06-05 11:40:50 CEST	session-cleanup-bunch
ssh-push-default	0 * * * * ?	2018-06-05 11:40:50 CEST	ssh-push-bunch
token-cleanup-default	0 0 0 ? * *	2018-06-05 11:40:51 CEST	token-cleanup-bunch
uuid-cleanup-default	0 0 * * * ?	2018-06-05 11:40:51 CEST	uuid-cleanup-bunch

Click a menu:SUSE Manager Schedules[Schedule name] to open its menu:Schedule Name[Basic Schedule Details] where you can disable it or change the frequency. Click btn:[Edit Schedule] to update the schedule with your settings. To delete a schedule, click btn:[Delete Schedule] in the upper right-hand corner.

WARNING Only disable or delete a schedule if you are absolutely certain this is necessary as they are essential for Uyuni to work properly.

If you click a bunch name, a list of runs of that bunch type and their status is displayed. Clicking the start time links takes you back to the menu:Schedule Name[Basic Schedule Details].

For example, the following predefined task bunches are scheduled by default and can be configured:

menu:channel-repodata-default:[]

(Re)generates repository metadata files.

menu:cleanup-data-default:[]

Cleans up stale package change log and monitoring time series data from the database.

menu:clear-taskologs-default:[]

Clears task engine (taskomatic) history data older than a specified number of days, depending on the job type, from the database.

menu:cobbler-sync-default:[]

Synchronizes distribution and profile data from Uyuni to Cobbler. For more information about autoinstallation powered by Cobbler, see **[Client-configuration › Autoinst-intro ›]**.

menu:compare-configs-default:[]

Compares configuration files as stored in configuration channels with the files stored on all configuration-enabled servers. To review comparisons, click menu:Systems[] tab and select the system of interest. Go to menu:Configuration[Compare Files]. For more information, see reference:systems/system-details/sd-configuration.pdf.

menu:cve-server-channels-default:[]

Updates internal pre-computed CVE data that is used to display results on the menu:Audit[CVE Audit] page. Search results in the menu:Audit[CVE Audit] page are updated to the last run of this schedule). For more information, see **[Reference › Audit ›]**.

menu:daily-status-default:[]

Sends daily report e-mails to relevant addresses. To learn more about how to configure notifications for specific users, see **[Reference › Users ›]**.

menu:errata-cache-default:[]

Updates internal patch cache database tables, which are used to look up packages that need updates for each server. Also, this sends notification emails to users that might be interested in certain patches. For more information about patches, see **[Reference › Patches ›]**.

menu:errata-queue-default:[]

Queues automatic updates (patches) for servers that are configured to receive them.

menu:kickstart-cleanup-default:[]

Cleans up stale kickstart session data.

menu:kickstartfile-sync-default:[]

Generates Cobbler files corresponding to Kickstart profiles created by the configuration wizard.

menu:mgr-forward-registration-default:[]

Synchronizes client registration data with SUSE Customer Center (SCC). By default, new, changed, or deleted client data are forwarded. To disable synchronization set in `/etc/rhn/rhn.conf`:

```
server.susemanager.forward_registration = 0
```

menu:mgr-sync-refresh-default:[]

The default time at which the start of synchronization with SUSE Customer Center (SCC) takes place (`mgr-sync-refresh`).

menu:minion-action-cleanup-default:[]

Deletes stale client action data from the file system. First it tries to complete any possibly unfinished actions by looking up the corresponding results; these results are stored in the Salt job cache. An unfinished action can occur if the server has missed the results of the action. For successfully completed actions it removes artifacts such as executed script files.

menu:package-cleanup-default:[]

Deletes stale package files from the file system.

menu:reboot-action-cleanup-default:[]

Any reboot actions pending for more than six hours are marked as failed and associated data is cleaned up in the database. For more information on scheduling reboot actions, see reference:systems/system-details/sd-provisioning.pdf.

menu:sandbox-cleanup-default:[]

Cleans up Sandbox configuration files and channels that are older than the *sandbox_lifetime* configuration parameter (3 days by default). Sandbox files are those imported from systems or files under development. For more information, see reference:systems/system-details/sd-configuration.pdf.

menu:session-cleanup-default:[]

Cleans up stale Web interface sessions, typically data that is temporarily stored when a user logs in and then closes the browser before logging out.

menu:ssh-push-default:[]

Prompts clients to check in with Uyuni via SSH if they are configured with a `SSH Push` contact method.

menu:token-cleanup-default:[]

Deletes expired repository tokens that are used by Salt clients to download packages and metadata.

Maintenance Windows

The maintenance windows feature in Uyuni allows you to schedule actions to occur during a scheduled maintenance window period. When you have created your maintenance window schedule, and applied it to a client, you are prevented from executing some actions outside of the specified period.

NOTE

Maintenance windows operate in a different way to system locking. System locks are switched on or off as required, while maintenance windows define periods of time when actions are allowed. Additionally, the allowed and restricted actions differ. For more information about system locks, see **[Client-configuration › System-locking ›]**.

Maintenance windows require both a calendar, and a schedule. The calendar defines the date and time of your maintenance window events, including recurring events, and must be in `ical` format. The schedule uses the events defined in the calendar to create the maintenance windows. You must create an `ical` file for upload, or link to an `ical` file to create the calendar, before you can create the schedule.

When you have created the schedule, you can assign it to clients that are registered to the Uyuni Server. Clients that have a maintenance schedule assigned cannot run restricted actions outside of maintenance windows.

Restricted actions significantly modify the client, and could potentially cause the client to stop running. Some examples of restricted actions are:

- Package installation

- Client upgrade

- Product migration

- Highstate application (for Salt clients)

Unrestricted actions are minor actions that are considered safe and are unlikely to cause problems on the client. Some examples of unrestricted actions are:

- Package profile update

- Hardware refresh

- Subscribing to software channels

Before you begin, you must create an `ical` file for upload, or link to an `ical` file to create the calendar. You can create `ical` files in your preferred calendaring tool, such as Microsoft Outlook, Google Calendar, or KOrganizer.

Procedure: Uploading a New Maintenance Calendar

1. In the Uyuni {webui}, navigate to menu:Schedule[Maintenance Windows > Calendars], and click btn:[Create].

2. In the `Calendar Name` section, type a name for your calendar.

3. Either provide a URL to your `ical` file, or upload the file directly.

4. Click btn:[Create Calendar] to save your calendar.

Procedure: Creating a New Schedule

1. In the Uyuni {webui}, navigate to menu:Schedule[Maintenance Windows > Schedules], and click btn:[Create].

2. In the `Schedule Name` section, type a name for your schedule.

3. OPTIONAL: If your `ical` file contains events that apply to more than one schedule, check `Multi`.

4. Select the calendar to assign to this schedule.

5. Click btn:[Create Schedule] to save your schedule.

Procedure: Assigning a Schedule to a Client

1. In the Uyuni {webui}, navigate to menu:Systems[Systems List], select the client to be assigned to a schedule, locate the `System Properties` panel, and click btn:[Edit These Properties]. Alternatively, you can assign clients through the system set manager by navigating to menu:Systems[System Set Manager] and using the menu:Misc[Maintenance Windows] tab.

2. In the `Edit System Details` page, locate the `Maintenance Schedule` field, and select the name of the schedule to be assigned.

3. Click btn:[Update Properties] to assign the maintenance schedule.

> **NOTE**
>
> When you assign a new maintenance schedule to a client, it is possible that the client might already have some restricted actions scheduled, and that these might now conflict with the new maintenance schedule. If this occurs, the {webui} displays an error and you cannot assign the schedule to the client. To resolve this, check the btn:[Cancel affected actions] option when you assign the schedule. This cancels any previously scheduled actions that conflict with the new maintenance schedule.

When you have created your maintenance windows, you can schedule restricted actions, such as package upgrades, to be performed during the maintenance window.

Procedure: Scheduling a Package Upgrade

1. In the Uyuni {webui}, navigate to menu:Systems[System List], select the client you want to upgrade, and go to the menu:Software[Packages > Upgrade] tab.

2. Select the package to upgrade from the list, and click btn:[Upgrade Packages].

3. In the `Maintenance Window` field, select which maintenance window the client should use to perform the upgrade.

4. Click btn:[Confirm] to schedule the package upgrade.

Maintenance Schedule Types

When you create a calendar, it contains a number of events, which can be either one-time events, or recurring events. Each event contains a `summary` field. If you want to create multiple maintenance schedules for one calendar, you can specify events for each using the `summary` field.

For example, you might like to create a schedule for production servers, and a different schedule for testing servers. In this case, you would specify `SUMMARY: Production Servers` on events for the production servers, and `SUMMARY: Testing Servers` on events for the testing servers.
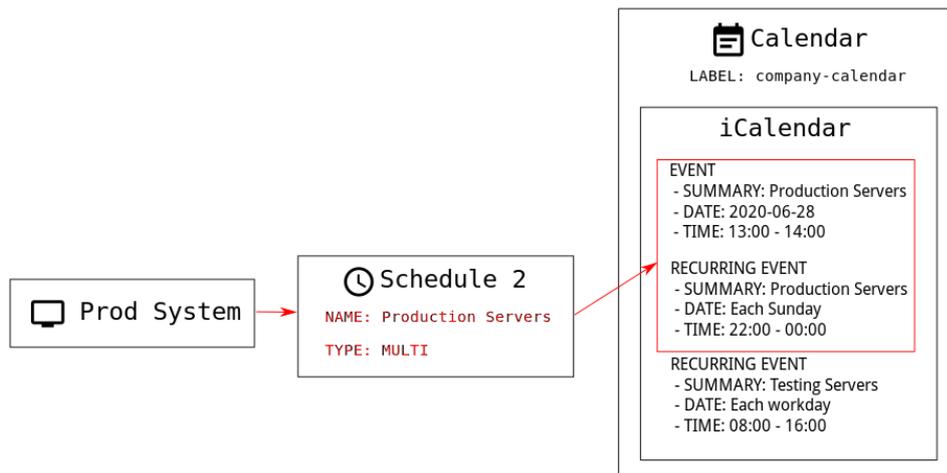
There are two types of schedule: single, or multi. If your calendar contains events that apply to more than one schedule, then you must select `multi`, and ensure you name the schedule according to the `summary` field you used in the calendar file.

Procedure: Creating a Multi Schedule

1. In the Uyuni {webui}, navigate to menu:Schedule[Maintenance Windows > Schedules], and click btn:[Create].

2. In the `Schedule Name` section, type the name for your schedule. Ensure it matches the `summary` field of the calendar.

3. Check the `Multi` option.

4. Select the calendar to assign to this schedule.

5. Click btn:[Create Schedule] to save your schedule.

6. To create the next schedule, click btn:[Create].

7. In the `Schedule Name` section, type the name for your second schedule. Ensure it matches the `summary` field of the second calendar.

8. Check the `Multi` option.

9. Click btn:[Create Schedule] to save your schedule.

10. Repeat for each schedule you need to create.

Restricted and Unrestricted Actions

This sections contains a complete list of restricted and unrestricted actions.

Restricted actions significantly modify the client, and could potentially cause the client to stop running. Restricted actions can only be run during a maintenance window. The restricted actions are:

- Package operations (for example, installing, updating, or removing packages)

- Patch updates

- Rebooting a client

- Rolling back transactions

- Configuration management changing tasks

- Applying a highstate (for Salt clients)

- Autoinstallation and reinstallation

- Remote commands

- Product migrations

- Cluster operations

> **NOTE**
>
> For Salt clients, it is possible to run remote commands directly at any time by navigating to menu:Salt[Remote Commands]. This applies whether or not the Salt client is in a maintenance window. For more information about remote commands, see **[Administration › Actions ›]**.

Unrestricted actions are minor actions that are considered safe and are unlikely to cause problems on the client. If an action is not restricted it is, by definition, unrestricted, and can be be run at any time.

Infrastructure Maintenance Tasks

If you work with scheduled downtime periods, you might find it difficult to remember all the things that you need to do before, during, and after that critical downtime of the Uyuni Server. Uyuni Server related systems such as Inter-Server Synchronization Slave Servers or Uyuni Proxies are also affected and have to be considered.

{suse} recommends you always keep your Uyuni infrastructure updated. That includes servers, proxies, and build hosts. If you do not keep the Uyuni Server updated, you might not be able to update some parts of your environment when you need to.

This section contains a checklist for your downtime period, with links to further information on performing each of the steps.

Server

1. Apply the latest updates. See **[Upgrade › Server-intro ›]**.

2. Upgrade to the latest service pack, if required.

3. Run `spacewalk-service status` and check whether all required services are up and running.

For information about database schema upgrades and PostgreSQL migrations, see **[Upgrade › Db-intro ›]**.

You can install updates using your package manager. For information on using {yast}, see https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-onlineupdate-you.html. For information

on using zypper, see https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-sw-cl.html#sec-zypper.

By default, several update channels are configured and enabled for the Uyuni Server. New and updated packages become available automatically.

Client Tools

When the server is updated consider updating some tools on the clients, too. Updating `salt-minion`, `zypper`, and other related management package on clients is not a strict requirement, but it is a best practice in general. For example, a maintenance update on the server might introduce a major new Salt version. Then Salt clients continue to work but might experience problems later on. To avoid this always update the `salt-minion` package when available. {suse} makes sure that `salt-minion` can always be updated safely.

Inter-Server Synchronization Slave Server

If you are using an inter-server synchronization slave server, update it after the Uyuni Server update is complete.

For more in inter-server synchronization, see **[Administration › Iss ›]**.

Monitoring Server

If you are using a monitoring server for Prometheus, update it after the Uyuni Server update is complete.

For more information on monitoring, see **[Administration › Monitoring ›]**.

Proxy

Proxies should be updated as soon as Uyuni Server updates are complete.

In general, running a proxy connected to a server on a different version is not supported. The only exception is for the duration of updates where it is expected that the server is updated first, so the proxy could run the previous version temporarily.

Especially if you are migrating from version 4.0 to 4.1, upgrade the server first, then any proxy.

For more information, see **[Upgrade › Proxy-intro ›]**.

Users

Uyuni Administrators can add new users, grant permissions, and deactivate or delete users. If you are managing a large number of users, you can assign users to system groups to manage permissions at a group level. You can also change the system defaults for the {webui}, including language and theme defaults.

> **NOTE**
> The `Users` menu is only available if you are logged in with the Uyuni administrator account.

To manage Uyuni users, navigate to menu:Users[User List > All] to see all users in your Uyuni Server. Each user in the list shows the username, real name, assigned roles, the date the user last signed in, and the current status of the user. Click btn:`Create User` to create a new user account. Click the username to go to the `User Details` page.

To add new users to your organization, click btn:[Create User], complete the details for the new user, and click btn:[Create Login].

Deactivate and Delete Accounts

You can deactivate or delete user accounts if they are no longer required. Deactivated user accounts can be reactivated at any time. Deleted user accounts are not visible, and cannot be retrieved.

Users can deactivate their own accounts. However, if users have an administrator role, the role must be removed before the account can be deactivated.

Deactivated users cannot log in to the Uyuni {webui} or schedule any actions. Actions scheduled by a user prior to their deactivation remain in the action queue. Deactivated users can be reactivated by Uyuni administrators.

Administrator Roles

Users can hold multiple administrator roles, and there can be more than one user holding any administrator role at any time. There must always be at least one active Uyuni Administrator.

To change a user's administrator roles, except for the Uyuni Administrator role, navigate to menu:Users[User List > All], select the user to change, and check or uncheck the administrator roles as required.

To change a user's Uyuni Administrator role, navigate to menu:Admin[Users] and check or uncheck `Uyuni Admin?` as required.

Table 1. User Administrator Role Permissions

Role Name	Description
System Group User	Standard role associated with all users.
Uyuni Administrator	Can perform all functions, including changing privileges of other users.
Organization Administrator	Manages activation keys, configurations, channels, and system groups.
Activation Key Administrator	Manages activation keys.

Role Name	Description
Image Administrator	Manages image profiles, builds, and stores.
Configuration Administrator	Manages system configuration.
Channel Administrator	Manages software channels, including making channels globally subscribable, and creating new channels.
System Group Administrator	Manages systems groups, including creating and deleting system groups, adding clients to existing groups, and managing user access to groups.

User Permissions and Systems

If you have created system groups to manage your clients, you can assign groups to users for them to manage.

To assign a user to a system group, navigate to menu:Users[User List], click the username to edit, and go to the `System Groups` tab. Check the groups to assign, and click btn:`Update Defaults`.

You can also select one or more default system groups for a user. When the user registers a new client, it is assigned to the chosen system group by default. This allows the user to immediately access the newly registered client.

To manage external groups, navigate to menu:Users[System Group Configuration], and go to the `External Authentication` tab. Click btn:[Create External Group] to create a new external group. Give the group a name, and assign it to the appropriate system group.

For more information about system groups, see [**Reference › Systems › **].

To see the individual clients a user can administer, navigate to menu:Users[User List], click the username to edit, and go to the `Systems` tab. To carry out bulk tasks, you can select clients from the list to add them to the system set manager.

For more information about the system set manager, see [**Client-configuration › System-set-manager ›**].

Users and Channel Permissions

You can assign users to software channels within your organization either as a subscriber that consumes content from channels, or as an administrator, who can manage the channels themselves.

To subscribe a user to a channel, navigate to menu:Users[User List], click the username to edit, and go to the menu:Channel Permissions[Subscription] tab. Check the channels to assign, and click btn:`Update Permissions`.

To grant a user channel management permissions, navigate to menu:Users[User List], click the username to edit, and go to the menu:Channel Permissions[Management] tab. Check the channels to assign, and click btn:Update Permissions.

Some channels in the list might not be subscribable. This is usually because of the users administrator status, or the channels global settings.

User Default Language

When you create a new user, you can choose which language to use for the {webui}. After a user has been created, you can change the language by navigating to menu:Home[My Preferences].

The default language is set in the `rhn.conf` configuration file. To change the default language, open the `/etc/rhn/rhn.conf` file and add or edit this line:

```
web.locale = <LANGCODE>
```

If the parameter is not set, the default language is `en_US`.

These languages are available in Uyuni:

Table 2. Available Language Codes

Language code	Language	Dialect
bn_IN	Bangla	India
ca	Catalan	
de	German	
en_US	English	United States
es	Spanish	
fr	French	
gu	Gujarati	
hi	Hindi	
it	Italian	
ja	Japanese	
ko	Korean	
pa	Punjabi	

Language code	Language	Dialect
pt	Portuguese	
pt_BR	Portuguese	Brazil
ru	Russian	
ta	Tamil	
zh_CN	Chinese	Mainland, Simplified
zh_TW	Chinese	Taiwan, Traditional

IMPORTANT Translations in {uyuni} are provided by the community, and could be incorrect or incomplete. Where a translation is not available, the {webui} defaults to English (en_US).

User Default Interface Theme

By default, the Uyuni {webui} uses the theme appropriate to the product you have installed. You can change the theme to reflect the {uyuni} or {susemgr} colors. The {susemgr} theme also has a dark option available.

You can change the default theme in the rhn.conf configuration file. To change the default theme, open the /etc/rhn/rhn.conf file and add or edit this line:

```
web.theme_default = <THEME>
```

Table 3. Available WebUI Themes

Theme Name	Colors	Style
susemanager-light	{susemgr}	Light
susemanager-dark	{susemgr}	Dark
uyuni	{uyuni}	Light

Backup and Restore

Back up your Uyuni installation regularly, to prevent data loss. Because Uyuni relies on a database as well as the installed program and configurations, it is important to back up all components of your installation. This chapter contains information on the files you need to back up, and introduces the smdba tool to manage database backups. It also contains information about restoring from your backups in the case of a system failure.

IMPORTANT	Regardless of the backup method you use, you must have available at least three times the amount of space your current installation uses. Running out of space can result in backups failing, so check this often.

Back up Uyuni

The most comprehensive method for backing up your Uyuni installation is to back up the relevant files and directories. This can save you time in administering your backup, and can be faster to reinstall and re-synchronize in the case of failure. However, this method requires significant disk space and could take a long time to perform the backup.

NOTE	If you want to only back up the required files and directories, use the following list. To make this process simpler, and more comprehensive, we recommend backing up the entire `/etc` and `/root` directories, not just the ones specified here. Some files only exist if you are actually using the related {susemgr} feature.

- `/etc/cobbler/`

- `/etc/dhcp.conf`

- `/etc/fstab` and any ISO mount points you require.

  If your UUID has changed, ensure you have updated the `fstab` entries accordingly.

- `/etc/rhn/`

- `/etc/salt`

- `/etc/sudoers`

- `/etc/sysconfig/rhn/`

- `/root/.gnupg/`

- `/root/.ssh`

  This file exists if you are using an SSH tunnel or SSH `push`. You also need to have saved a copy of the `id-susemanager` key.

- `/root/ssl-build/`

- `/srv/formula_metadata`

- `/srv/pillar`

- `/srv/salt`

- `/srv/susemanager`

- `/srv/tftpboot/`

- `/srv/www/cobbler`

- `/srv/www/htdocs/pub/`

- `/srv/www/os-images`

- `/var/cache/rhn`

- `/var/cache/salt`

- `/var/lib/cobbler/`

- `/var/lib/cobbler/templates/` (before version 4.0 it is `/var/lib/rhn/kickstarts/`)

- `/var/lib/Kiwi`

- `/var/lib/rhn/`

- `/var/run/pgsql/`

- `/var/lib/salt/`

- `/var/run/salt/`

- `/var/spacewalk/`

- Any directories containing custom data such as scripts, Kickstart or AutoYaST profiles, and custom RPMs.

> **IMPORTANT**   You also need to back up your database, which you can do with the `smdba` tool.

Procedure: Restoring from a Manual Backup

1. Re-install the Uyuni Server. For more information, see **[Installation › Install-server-unified ›]**.

2. Set up the Uyuni Server with `yast2 susemanager_setup`. For more information, see **[Installation › Server-setup ›]**.

3. Re-synchronize your Uyuni repositories using either the Uyuni {webui}, or with the `mgr-sync` tool at the command prompt. You can choose to re-register your product, or skip the registration and SSL certificate generation sections.

4. Re-install the `/root/ssl-build/rhn-org-httpd-ssl-key-pair-MACHINE_NAME-VER-REL.noarch.rpm` package.

5. Schedule the re-creation of search indexes next time the `rhn-search` service is started. This command produces only debug messages, it does not produce error messages:

```
rhn-search cleanindex
```

6. Check whether you need to restore `/var/spacewalk/packages/`. If `/var/spacewalk/packages/` was not in your backup, you need to restore it. If the source repository is available, you can restore `/var/spacewalk/packages/` with a complete channel

synchronization:

```
mgr-sync refresh --refresh-channels
```

Administering the Database with smdba

The smdba tool is used for managing a local PostgreSQL database. It allows you to back up and restore your database, and manage backups. It can also be used to check the status of your database, and perform administration tasks, such as restarting.

The smdba tool works with local PostgreSQL databases only, it does not work with remotely accessed databases, or Oracle databases.

IMPORTANT

The smdba tool requires sudo access, to execute system changes. Ensure you have enabled sudo access for the admin user before you begin, by checking the /etc/sudoers file for this line:

```
admin   ALL=(postgres) /usr/bin/smdba
```

Check the runtime status of your database with:

```
smdba db-status
```

This command returns either online or offline, for example:

```
Checking database core...        online
```

Starting and stopping the database can be performed with:

```
smdba db-start
```

And:

```
smdba db-stop
```

Database Backup with smdba

The smdba tool performs a continuous archiving backup. This backup method combines a log of every change made to the database during the current session, with a series of more traditional backup files. When a crash occurs, the database state is first restored from the most recent backup file on disk, then the

log of the current session is replayed exactly, to bring the database back to a current state. A continuous archiving backup with smdba is performed with the database running, so there is no need for downtime.

This method of backing up is stable and generally creates consistent snapshots, however it can take up a lot of storage space. Ensure you have at least three times the current database size of space available for backups. You can check your current database size by navigating to /var/lib/pgsql/ and running df -h.

The smdba tool also manages your archives, keeping only the most recent backup, and the current archive of logs. The log files can only be a maximum file size of 16 MB, so a new log file is created when the files reach this size. Every time you create a new backup, previous backups are purged to release disk space. We recommend you use cron to schedule your smdba backups to ensure that your storage is managed effectively, and you always have a backup ready in case of failure.

Perform a Manual Database Backup

The smdba tool can be run directly from the command line. We recommend you run a manual database backup immediately after installation, or if you have made any significant changes to your configuration.

NOTE
When smdba is run for the first time, or if you have changed the location of the backup, it needs to restart your database before performing the archive. This results in a small amount of downtime. Regular database backups do not require any downtime.

Procedure: Performing a Manual Database Backup

1. Allocate permanent storage space for your backup. This example uses a directory located at /var/spacewalk/. This becomes a permanent target for your backup, so ensure it remains accessible by your server at all times.

2. In your backup location, create a directory for the backup:

   ```
   sudo -u postgres mkdir /var/spacewalk/db-backup
   ```

   Or, as root:

   ```
   install -d -o postgres -g postgres -m 700 /var/spacewalk/db-backup
   ```

3. Ensure you have the correct permissions set on the backup location:

   ```
   chown postgres:postgres /var/spacewalk/db-backup
   ```

4. To create a backup for the first time, run the smdba backup-hot command with the enable option set. This creates the backup in the specified directory, and, if necessary, restart the database:

```
smdba backup-hot --enable=on --backup-dir=/var/spacewalk/db-backup
```

This command produces debug messages and finishes sucessfully with the output:

```
INFO: Finished
```

5. Check that the backup files exist in the `/var/spacewalk/db-backup` directory, to ensure that your backup has been successful.

Scheduling Automatic Backups

You do not need to shut down your system to perform a database backup with `smdba`. However, because it is a large operation, database performance can slow down while the backup is running. We recommend you schedule regular database backups for a low-traffic period, to minimize disruption.

> **IMPORTANT** Ensure you have at least three times the current database size of space available for backups. You can check your current database size by navigating to `/var/lib/pgsql/` and running `df -h`.

Procedure: Scheduling Automatic Backups

1. Create a directory for the backup, and set the appropriate permissions (as root):

```
install -m 700 -o postgres -g postgres /var/spacewalk/db-backup
```

2. Open `/etc/cron.d/db-backup-mgr`, or create it if it does not exist, and add the following line to create the cron job:

```
0 2 * * * root /usr/bin/smdba backup-hot --enable=on --backup-dir=/var/spacewalk/db
-backup
```

3. Check the backup directory regularly to ensure the backups are working as expected.

Restore from Backup

The `smdba` tool can be used to restore from backup in the case of failure.

Procedure: Restoring from Backup

1. Shut down the database:

```
smdba db-stop
```

2. Start the restore process and wait for it to complete:

```
smdba backup-restore start
```

3. Restart the database:

```
smdba db-start
```

4. Check if there are differences between the RPMs and the database.

```
spacewalk-data-fsck
```

Archive Log Settings

Archive logging allows the database management tool `smdba` to perform hot backups. In Uyuni with an embedded database, archive logging is enabled by default.

PostgreSQL maintains a limited number of archive logs. Using the default configuration, approximately 64 files with a size of 16 MiB are stored.

Creating a user and syncing the channels:

- SLES12-SP2-Pool-x86_64

- SLES12-SP2-Updates-x86_64

- SLE-Manager-Tools12-Pool-x86_64-SP2

- SLE-Manager-Tools12-Updates-x86_64-SP2

PostgreSQL generates an additional roughly 1 GB of data. So it is important to think about a backup strategy and create a backups in a regular way.

Archive logs are stored at `/var/lib/pgsql/data/pg_xlog/` (postgresql).

Retrieve an Overview of Occupied Database Space

Database administrators may use the subcommand `space-overview` to get a report about occupied table spaces, for example:

```
smdba space-overview
```

outputs:

```
SUSE Manager Database Control. Version 1.5.2
Copyright (c) 2012 by SUSE Linux Products GmbH

Tablespace  | Size (Mb) | Avail (Mb) | Use %
------------+-----------+------------+------
postgres    | 7         | 49168      | 0.013
susemanager | 776       | 48399      | 1.602
```

The `smdba` command is available for PostgreSQL. For a more detailed report, use the `space-tables` subcommand. It lists the table and its size, for example:

```
smdba space-tables
```

outputs:

```
SUSE Manager Database Control. Version 1.5.2
Copyright (c) 2012 by SUSE Linux Products GmbH

Table                              | Size
-----------------------------------+-----------
public.all_primary_keys            | 0 bytes
public.all_tab_columns             | 0 bytes
public.allserverkeywordsincereboot | 0 bytes
public.dblink_pkey_results         | 0 bytes
public.dual                        | 8192 bytes
public.evr_t                       | 0 bytes
public.log                         | 32 kB
...
```

Move the Database

It is possible to move the database to another location. For example, if database storage space is running low.

Procedure: Moving the Database

1. The default storage location for Uyuni is `/var/lib/pgsql/`. If you would like to move it, for example to `/storage/postgres/`, proceed as follows.

2. At the command prompt, as root, stop the running database:

   ```
   rcpostgresql stop
   ```

   Shut down the running spacewalk services:

   ```
   spacewalk-service stop
   ```

3. Copy the current working directory structure with `cp` using the `-a, --archive` option. For example:

```
cp --archive /var/lib/pgsql/ /storage/postgres/
```

This command copies the contents of `/var/lib/pgsql/` to `/storage/postgres/pgsql/`.

> **IMPORTANT**
>
> The contents of the `/var/lib/pgsql` directory needs to remain the same, otherwise the Uyuni database may malfunction. You also should ensure that there is enough available disk space.

4. Mount the new database directory:

```
mount /storage/postgres/pgsql
```

5. Make sure ownership is `postgres:postgres` and not `root:root` by changing to the new directory and running these commands:

```
cd /storage/postgres/pgsql/
ls -l
```

Outputs:

```
total 8
drwxr-x---  4 postgres postgres   47 Jun  2 14:35 ./
```

6. Add the new database mount location to your servers fstab by editing `etc/fstab`.

7. Start the database with:

```
rcpostgresql start
```

8. Start the spacewalk services:

```
spacewalk-service start
```

Recover From a Crashed Root Partition

If your root partition has crashed, you can restart the Uyuni Server with some additional steps. This section assumes you have setup your server using separate partitions for the database and for channels, mounted at `/var/lib/pgsql` and `/var/spacewalk/`.

Procedure: Recovering from a Crashed Root Partition

1. Install Uyuni. Do not mount the `/var/spacewalk` and `/var/lib/pgsql` partitions. Wait for the

installation to complete before going on to the next step.

2. Shut down the services with `spacewalk-service shutdown`.

3. Shut down the database with `rcpostgresql stop`.

4. Mount `/var/spacewalk` and `/var/lib/pgsql` partitions.

5. Restore the directories listed in Back up Uyuni.

6. Start the spacewalk services with `spacewalk-services start`.

7. Start the database with `rcpostgresql start`.

Uyuni should now operate normally without loss of your database or synced channels.

Database Connection Information

You can set information for connecting to the Uyuni database by adding or editing these variable in `/etc/rhn/rhn.conf`:

```
db_backend = postgresql
db_user = susemanager
db_password = susemanager
db_name = susemanager
db_host = localhost
db_port = 5432
db_ssl_enabled =
```

Managing Disk Space

Running out of disk space can have a severe impact on the Uyuni database and file structure which, in some cases, is not recoverable.

Uyuni monitors some directories for free disk space. You can modify which directories are monitored, and the warnings that are created. All settings are configured in the `/etc/rhn/rhn.conf` configuration file.

Monitored Directories

By default, Uyuni monitors these directories:

- `/var/lib/pgsql`
- `/var/spacewalk`
- `/var/cache`
- `/srv`

You can change which directories are monitored with the `spacecheck_dirs` parameter. You can

specify multiple directories by separating them with a space.

For example:

```
spacecheck_dirs = /var/lib/pgsql /var/spacewalk /var/cache /srv
```

Thresholds

By default, Uyuni creates a warning mail when a monitored directory has less than 10% of total space available. A critical alert is created when a monitored directory falls below 5% space available.

You can change these alert thresholds with the `spacecheck_free_alert` and `spacecheck_free_critical` parameters.

For example:

```
spacecheck_free_alert = 10
spacecheck_free_critical = 5
```

Shut Down Services

By default, Uyuni shuts down the spacewalk services when the critical alert threshold is reached.

You can change this behavior with the `spacecheck_shutdown` parameter. A value of `true` enables the shut down feature. Any other value disables it.

For example:

```
spacecheck_shutdown = true
```

Disable Space Checking

The space checking tool is enabled by default. You can disable it entirely with these commands:

```
systemctl stop spacewalk-diskcheck.timer
systemctl disable spacewalk-diskcheck.timer
```

Using `mgr-sync`

The `mgr-sync` tool is used at the command prompt. It provides functions for using Uyuni that are not always available in the {webui}. The primary use of `mgr-sync` is to connect to the {scc}, retrieve product and package information, and prepare channels for synchronization with the Uyuni Server.

This tool is designed for use with a {suse} support subscription. It is not required for open source distributions, including {opensuse}, {centos}, and {ubuntu}.

The available commands and arguments for `mgr-sync` are listed in this table. Use this syntax for `mgr-sync` commands:

```
mgr-sync [-h] [--version] [-v] [-s] [-d {1,2,3}] {list,add,refresh,delete}
```

Table 4. mgr-sync Commands

Command	Description	Example Use
list	List channels, organization credentials, or products	`mgr-sync list channels`
add	Add channels, organization credentials, or products	`mgr-sync add channel <channel_name>`
refresh	Refresh the local copy of products, channels, and subscriptions	`mgr-sync refresh`
delete	Delete existing SCC organization credentials from the local system	`mgr-sync delete credentials`
sync	Synchronize specified channel or ask for it when left blank	`mgr-sync sync channel <channel_name>`

To see the full list of options specific to a command, use this command:

```
mgr-sync <command> --help
```

Table 5. mgr-sync Optional Arguments

Option	Abbreviated option	Description	Example Use
help	h	Display the command usage and options	`mgr-sync --help`
version	N/A	Display the currently installed version of `mgr-sync`	`mgr-sync --version`
verbose	v	Provide verbose output	`mgr-sync --verbose refresh`
store-credentials	s	Store credentials a local hidden file	`mgr-sync --store-credentials`

Option	Abbreviated option	Description	Example Use
debug	d	Log additional debugging information. Requires a level of 1, 2, 3. 3 provides the highest amnount of debugging information	`mgr-sync -d 3 refresh`
no-sync	N/A	Use with the `add` command to add products or channels without beginning a synchronization	`mgr-sync --no-sync add <channel_name>`

Logs for `mgr-sync` are located in:

- `/var/log/rhn/mgr-sync.log`
- `/var/log/rhn/rhn_web_api.log`

Security

Set up a Client to Master Validation Fingerprint

In highly secure network configurations you may wish to ensure your Salt clients are connecting a specific master. To set up validation from client to master enter the master's fingerprint within the `/etc/salt/minion` configuration file.

See the following procedure:

1. On the master, at the command prompt, as root, use this command to find the `master.pub` fingerprint:

   ```
   salt-key -F master
   ```

   On your client, open the `/etc/salt/minion` configuration file. Uncomment the following line and enter the master's fingerprint replacing the example fingerprint:

   ```
   master_finger: 'ba:30:65:2a:d6:9e:20:4f:d8:b2:f3:a7:d4:65:11:13'
   ```

2. Restart the salt-minion service:

   ```
   # systemctl restart salt-minion
   ```

For information on configuring security from a client, see https://docs.saltstack.com/en/latest/ref/configuration/minion.html.

Signing Repository Metadata

You require a custom GPG key to be able to sign repository metadata.

Procedure: Generating a Custom GPG Key

1. As the root user, use the `gpg` command to generate a new key:

```
gpg --gen-key
```

2. At the prompts, select `RSA` as the key type, with a size of 2048 bits, and select an appropriate expiry date for your key. Check the details for your new key, and type `y` to confirm.

3. At the prompts, enter a name and email address to be associated with your key. You can also add a comment to help you identify the key, if desired. When you are happy with the user identity, type `O` to confirm.

4. At the prompt, enter a passphrase to protect your key.

5. The key should be automatically added to your keyring. You can check by listing the keys in your keyring:

```
gpg --list-keys
```

6. Add the password for your keyring to the `/etc/rhn/signing.conf` configuration file, by opening the file in your text editor and adding this line:

```
GPGPASS="password"
```

You can manage metadata signing on the command line using the `mgr-sign-metadata-ctl` command.

Procedure: Enabling Metadata Signing

1. You need to know the short identifier for the key to use. You can list your available public keys in short format:

```
gpg --keyid-format short --list-keys
...
pub   rsa2048/3E7BFE0A 2019-04-02 [SC] [expires: 2021-04-01]
      A43F9EC645ED838ED3014B035CFA51BF3E7BFE0A
uid         [ultimate] SUSE Manager
sub   rsa2048/118DE7FF 2019-04-02 [E] [expires: 2021-04-01]
```

2. Enable metadata signing with the `mgr-sign-metadata-ctl` command:

```
mgr-sign-metadata-ctl enable 3E7BFE0A
OK. Found key 3E7BFE0A in keyring.
DONE. Set key 3E7BFE0A in /etc/rhn/signing.conf.
DONE. Enabled metadata signing in /etc/rhn/rhn.conf.
DONE. Exported key 4E2C3DD8 to /srv/susemanager/salt/gpg/mgr-keyring.gpg.
DONE. Exported key 4E2C3DD8 to /srv/www/htdocs/pub/mgr-gpg-pub.key.
NOTE. For the changes to become effective run:
    mgr-sign-metadata-ctl regen-metadata
```

3. You can check that your configuration is correct with this command:

```
mgr-sign-metadata-ctl check-config
```

4. Restart the services and schedule metadata regeneration to pick up the changes:

```
mgr-sign-metadata-ctl regen-metadata
```

You can also use the `mgr-sign-metadata-ctl` command to perform other tasks. Use `mgr-sign-metadata-ctl --help` to see the complete list.

Repository metadata signing is a global option. When it is enabled, it is enabled on all software channels on the server. This means that all clients connected to the server need to trust the new GPG key to be able to install or update packages.

Procedure: Importing GPG keys on Clients

1. For RPM-based client systems, use these remote commands:

```
rpm --import http://server.example.com/pub/mgr-gpg-pub.key
```

2. For Ubuntu clients, you need to reassign the channels, which automatically pick up the new GPG key. You can do this through the Uyuni {webui}, or from the command line on the server with this command:

```
salt <ubuntu-client> state.apply channels
```

3. OPTIONAL: For Salt clients, you might prefer to use a state to manage your GPG keys.

Mirror Source Packages

If you build your own packages locally, or if you require the source code for your packages for legal reasons, it is possible to mirror the source packages on Uyuni Server.

> **NOTE** Mirroring source packages can consume a significant amount of disk space.

Procedure: Mirroring Source Packages

1. Open the `/etc/rhn/rhn.conf` configuration file, and add this line:

```
server.sync_source_packages = 1
```

2. Restart the Spacewalk service to pick up the changes:

```
spacewalk-service restart
```

Currently, this feature can only be enabled globally for all repositories. It is not possible to select individual repositories for mirroring.

When this feature has been activated, the source packages become available in the Uyuni {webui} after the next repository synchronization. They are shown as sources for the binary package, and can be downloaded directly from the {webui}. Source packages cannot be installed on clients using the {webui}.

System Security with OpenSCAP

Uyuni uses OpenSCAP to audit clients. It allows you to schedule and view compliance scans for any client.

About SCAP

The Security Certification and Authorization Package (SCAP) is a standardized compliance checking solution for enterprise-level Linux infrastructures. It is a line of specifications maintained by the National Institute of Standards and Technology (NIST) for maintaining system security for enterprise systems.

SCAP was created to provide a standardized approach to maintaining system security, and the standards that are used continually change to meet the needs of the community and enterprise businesses. New specifications are governed by NIST's SCAP Release cycle to provide a consistent and repeatable revision work flow. For more information, see http://scap.nist.gov/timeline.html.

Uyuni uses OpenSCAP to implement the SCAP specifications. OpenSCAP is an auditing tool that utilizes the Extensible Configuration Checklist Description Format (XCCDF). XCCDF is a standard way of expressing checklist content and defines security checklists. It also combines with other specifications such as Common Platform Enumeration (CPE), Common Configuration Enumeration (CCE), and Open Vulnerability and Assessment Language (OVAL), to create a SCAP-expressed checklist that can be processed by SCAP-validated products.

OpenSCAP verifies the presence of patches by using content produced by the {suse} Security Team. OpenSCAP checks system security configuration settings and examines systems for signs of compromise by using rules based on standards and specifications. For more information about the {suse} Security

Team, see https://www.suse.com/support/security.

Prepare Clients for an SCAP Scan

Before you begin, you need to prepare your client systems for SCAP scanning.

| NOTE | OpenSCAP auditing is not available on Salt clients that use the SSH contact method. |

| IMPORTANT | Scanning clients can consume a lot of memory and compute power on the client being scanned. For {redhat} clients, ensure you have at least 2 GB of RAM available on each client to be scanned. |

For traditional and Salt clients, install the OpenSCAP scanner and the SCAP Security Guide (content) packages before you begin. Depending on the operating system, these packages are included either on the base operating system, or in the Uyuni Client Tools.

The table below lists the packages you need depending on your client operating system:

Table 6. OpenSCAP packages

Operating system	Scanner	Content
SLES	openscap-utils	scap-security-guide
openSUSE	openscap-utils	scap-security-guide
RHEL	openscap-utils	scap-security-guide-redhat
CentOS	openscap-utils	scap-security-guide-redhat
Oracle Linux	openscap-utils	scap-security-guide-redhat
Ubuntu	libopenscap8	scap-security-guide-ubuntu
Debian	libopenscap8	scap-security-guide-debian

RHEL 7 and compatible systems provide a `scap-security-guide` package, which contains outdated contents. You are advised to use the `scap-security-guide-redhat` package you will find in the Uyuni Client Tools.

OpenSCAP Content Files

OpenSCAP uses SCAP content files to define test rules. These content files are created based on the XCCDF or OVAL standards. In addition to the SCAP Security Guide, you can download publicly available content files and customize it to your requirements. You can install the SCAP Security Guide package for default content file templates. Alternatively, if you are familiar with XCCDF or OVAL, you can create your own content files.

WARNING	We recommend you use templates to create your SCAP content files. If you create and use your own custom content files, you do so at your own risk. If your system becomes damaged through the use of custom content files, you might not be supported by {suse}.

When you have created your content files, you need to transfer the file to the client. You can do this in the same way as you move any other file, using physical storage media, or across a network with Salt (for example, salt-cp or the Salt File Server), `ftp` or `scp`.

We recommend that you create a package to distribute content files to clients that you are managing with Uyuni. Packages can be signed and verified to ensure their integrity. For more information, see **[Administration › Custom-channels ›]**.

Find OpenSCAP profiles

Different operating systems make available different OpenSCAP content files and profiles. One content file may contain more than one profile.

On RPM-based operating systems, use this command to determine the location of the available SCAP files:

```
rpm -ql <scap-security-guide-package-name-from-table>
```

On DEB-based operating systems, use this command to determine the location of the available SCAP files:

```
dpkg -L <scap-security-guide-package-name-from-table>
```

When you have identified one SCAP content file that suits your needs, list profiles available on the client:

```
oscap info ./usr/share/xml/scap/ssg/content/ssg-sle15-ds-1.2.xml
Document type: Source Data Stream
Imported: 2021-03-24T18:14:45

Stream: scap_org.open-scap_datastream_from_xccdf_ssg-sle15-xccdf-1.2.xml
Generated: (null)
Version: 1.2
Checklists:
        Ref-Id: scap_org.open-scap_cref_ssg-sle15-xccdf-1.2.xml
                Status: draft
                Generated: 2021-03-24
                Resolved: true
                Profiles:
                        Title: CIS SUSE Linux Enterprise 15 Benchmark
                                Id: xccdf_org.ssgproject.content_profile_cis
                        Title: Standard System Security Profile for SUSE Linux Enterprise 15
                                Id: xccdf_org.ssgproject.content_profile_standard
                        Title: DISA STIG for SUSE Linux Enterprise 15
                                Id: xccdf_org.ssgproject.content_profile_stig
                Referenced check files:
                        ssg-sle15-oval.xml
                                system: http://oval.mitre.org/XMLSchema/oval-definitions-5
                        ssg-sle15-ocil.xml
                                system: http://scap.nist.gov/schema/ocil/2

https://ftp.suse.com/pub/projects/security/oval/suse.linux.enterprise.15.xml
                                system: http://oval.mitre.org/XMLSchema/oval-definitions-5
Checks:
        Ref-Id: scap_org.open-scap_cref_ssg-sle15-oval.xml
        Ref-Id: scap_org.open-scap_cref_ssg-sle15-ocil.xml
        Ref-Id: scap_org.open-scap_cref_ssg-sle15-cpe-oval.xml
Dictionaries:
        Ref-Id: scap_org.open-scap_cref_ssg-sle15-cpe-dictionary.xml
```

Take a note of the file paths and profiles for performing the scan.

Perform an Audit Scan

When you have installed or transferred your content files, you can perform audit scans. Audit scans can be triggered using the Uyuni {webui}. You can also use the Uyuni API to schedule regular scans.

Procedure: Running an Audit Scan from the {webui}

1. In the Uyuni {webui}, navigate to menu:Systems[Systems List] and select the client you want to scan.

2. Navigate to the `Audit` tab, and the `Schedule` subtab.

3. In the `Path to XCCDF Document` field, enter the parameters for the SCAP template and profile you want to use on the client. For example:

```
Command: /usr/bin/oscap xccdf eval
Command-line arguments: --profile xccdf_org.ssgproject.content_profile_standard
Path to XCCDF document: /usr/share/xml/scap/ssg/content/ssg-sle15-ds-1.2.xml
```

1. The scan runs at the client's next scheduled synchronization.

> The XCCDF content file is validated before it is run on the remote system. If the content file includes invalid arguments, the test fails.

Procedure: Running an Audit Scan from the API

1. Before you begin, ensure that the client to be scanned has Python and XML-RPC libraries installed.

2. Choose an existing script or create a script for scheduling a system scan through `system.scap.scheduleXccdfScan`. For example:

```
#!/usr/bin/python
client = xmlrpclib.Server('https://server.example.com/rpc/api')
key = client.auth.login('username', 'password')
client.system.scap.scheduleXccdfScan(key, <1000010001>,
    '<path_to_xccdf_file.xml>',
    '--profile <profile_name>')
```

In this example: * `<1000010001>` is the system ID (sid). * `<path_to_xccdf_file.xml>` is the path to the content file location on the client. For example, `/usr/share/xml/scap/ssg/content/ssg-sle15-ds-1.2.xml`. * `<profile_name>` is an additional argument for the `oscap` command. For example, use `united_states_government_configuration_baseline` (USGCB).

3. Run the script on the client you want to scan, from the command prompt.

Scan Results

Information about the scans you have run is in the Uyuni {webui}. Navigate to to menu:Audit[OpenSCAP > All Scans] for a table of results. For more information about the data in this table, see **[Reference › Audit ›]**.

To ensure that detailed information about scans is available, you need to enable it on the client. In the Uyuni {webui}, navigate to menu:Admin[Organizations] and click on the organization the client is a part of. Navigate to the `Configuration` tab, and check the `Enable Upload of Detailed SCAP Files` option. When enabled, this generates an additional HTML file on every scan, which contains extra information. The results show an extra line similar to this:

```
Detailed Results: xccdf-report.html xccdf-results.xml scap-yast2sec-oval.xml.result.xml
```

To retrieve scan information from the command line, use the `spacewalk-report` command:

```
spacewalk-report system-history-scap
spacewalk-report scap-scan
spacewalk-report scap-scan-results
```

You can also use the Uyuni API to view results, with the `system.scap` handler.

Remediation

Remediation bash scripts and Ansible playbooks are provided in the same SCAP Security Guide packages to harden the client systems. For example:

bash scripts

```
/usr/share/scap-security-guide/bash/sle15-script-cis.sh
/usr/share/scap-security-guide/bash/sle15-script-standard.sh
/usr/share/scap-security-guide/bash/sle15-script-stig.sh
```

Ansible playbooks

```
/usr/share/scap-security-guide/ansible/sle15-playbook-cis.yml
/usr/share/scap-security-guide/ansible/sle15-playbook-standard.yml
/usr/share/scap-security-guide/ansible/sle15-playbook-stig.yml
```

You can run them using remote commands or with Ansible, after enabling Ansible in the client system.

Auditing

In Uyuni, you can keep track of your clients through a series of auditing tasks. You can check that your clients are up to date with all public security patches (CVEs), perform subscription matching, and use OpenSCAP to check for specification compliance.

In the Uyuni {webui}, navigate to `Audit` to perform auditing tasks.

CVE Audits

A CVE (common vulnerabilities and exposures) is a fix for a publicly known security vulnerability.

> **IMPORTANT**   You must apply CVEs to your clients as soon as they become available.

Each CVE contains an identification number, a description of the vulnerability, and links to further information. CVE identification numbers use the form `CVE-YEAR-XXXX`.

In the Uyuni {webui}, navigate to menu:Audit[CVE Audit] to see a list of all clients and their current patch status.

By default, the CVE data is updated at 2300 every day. We recommend that before you begin a CVE audit you refresh the data to ensure you have the latest patches.

Procedure: Updating CVE Data

1. In the Uyuni {webui}, navigate to menu:Admin[Task Schedules] and select the `cve-server-channels-default` schedule.

2. Click btn:[cve-server-channels-bunch].

3. Click btn:[Single Run Schedule] to schedule the task. Allow the task to complete before continuing with the CVE audit.

Procedure: Verifying Patch Status

1. In the Uyuni {webui}, navigate to menu:Audit[CVE Audit].

2. OPTIONAL: To check the patch status for a particular CVE, type the CVE identifier in the `CVE Number` field.

3. Select the patch statuses you want to look for, or leave all statuses checked to look for all.

4. Click btn:[Audit Servers] to check all systems, or click btn:[Audit Images] to check all images.

For more information about the patch status icons used on this page, see **[Reference › Audit ›]**.

For each system, the `Next Action` column provides information about what you need to do to address vulnerabilities. If applicable, a list of candidate channels or patches is also given. You can also assign systems to a `System Set` for further batch processing.

You can use the Uyuni API to verify the patch status of your clients. Use the `audit.listSystemsByPatchStatus` API method. For more information about this method, see the Uyuni API Guide.

CVE Status

The CVE status of clients is usually either `affected`, `not affected`, or `patched`. These statuses are based only on the information that is available to Uyuni.

Within Uyuni, these definitions apply:

System affected by a certain vulnerability

A system which has an installed package with version lower than the version of the same package in a relevant patch marked for the vulnerability.

System not affected by a certain vulnerability

A system which has no installed package that is also in a relevant patch marked for the vulnerability.

System patched for a certain vulnerability

A system which has an installed package with version equal to or greater than the version of the same package in a relevant patch marked for the vulnerability.

Relevant patch

A patch known by Uyuni in a relevant channel.

Relevant channel

A channel managed by Uyuni, which is either assigned to the system, the original of a cloned channel which is assigned to the system, a channel linked to a product which is installed on the system or a past or future service pack channel for the system.

NOTE	Because of the definitions used within Uyuni, CVE audit results might be incorrect in some circumstances. For example, unmanaged channels, unmanaged packages, or non-compliant systems might report incorrectly.

Generate Reports

The `spacewalk-report` command is used to produce a variety of reports. These reports can be helpful for taking inventory of your subscribed systems, users, and organizations. Using reports is often simpler than gathering information manually from the {susemgr} {webui}, especially if you have many systems under management.

To generate reports, you must have the `spacewalk-reports` package installed.

The `spacewalk-report` command allows you to organize and display reports about content, systems, and user resources across Uyuni.

You can generate reports on:

1. System Inventory: list all the systems registered to Uyuni.

2. Patches: list all the patches relevant to the registered systems. You can sort patches by severity, as well as the systems that apply to a particular patch.

3. Users: list all registered users and any systems associated with a particular user.

To get the report in CSV format, run this command at the command prompt on the server:

```
spacewalk-report <report_name>
```

This table lists the available reports:

Table 7. `spacewalk-report` *Reports*

Report	Invoked as	Description
Actions	`actions`	All actions.
Activation Keys	`activation-keys`	All activation keys, and the entitlements, channels, configuration channels, system groups, and packages associated with them.
Activation Keys: Channels	`activation-keys-channels`	All activation keys and the entities associated with each key.

Report	Invoked as	Description
Activation Keys: Configuration	`activation-keys-config`	All activation keys and the configuration channels associated with each key.
Activation Keys: Server Groups	`activation-keys-groups`	All activation keys and the system groups associated with each key.
Activation Keys: Packages	`activation-keys-packages`	All activation keys and the packages each key can deploy.
Channel Packages	`channel-packages`	All packages in a channel.
Channel Report	`channels`	Detailed report of a given channel.
Cloned Channel Report	`cloned-channels`	Detailed report of cloned channels.
Configuration Files	`config-files`	All configuration file revisions for all organizations, including file contents and file information.
Latest Configuration Files	`config-files-latest`	The most recent configuration file revisions for all organizations, including file contents and file information.
Custom Channels	`custom-channels`	Channel metadata for all channels owned by specific organizations.
Custom Info	`custom-info`	Client custom information.
Patches in Channels	`errata-channels`	All patches in channels.
Patches Details	`errata-list`	All patches that affect registered clients.
All patches	`errata-list-all`	All patches.
Patches for Clients	`errata-systems`	Applicable patches and any registered clients that are affected.
Host Guests	`host-guests`	Host and guests mapping.
Inactive Clients	`inactive-systems`	Inactive clients.
System Inventory	`inventory`	Clients registered to the server, together with hardware and software information.

Report	Invoked as	Description
Kickstart Scripts	`kickstart-scripts`	All kickstart scripts, with details.
Kickstart Trees	`kickstartable-trees`	Kickstartable trees.
All Upgradable Versions	`packages-updates-all`	All newer package versions that can be upgraded.
Newest Upgradable Version	`packages-updates-newest`	Newest package versions that can be upgraded.
Proxy Overview	`proxies-overview`	All proxies and the clients registered to each.
Repositories	`repositories`	All repositories, with their associated SSL details, and any filters.
Result of SCAP	`scap-scan`	Result of OpenSCAP `sccdf` evaluations.
Result of SCAP	`scap-scan-results`	Result of OpenSCAP `sccdf` evaluations, in a different format.
System Data	`splice-export`	Client data needed for splice integration.
System Currency	`system-currency`	Number of available patches for each registered client.
System Extra Packages	`system-extra-packages`	All packages installed on all clients that are not available from channels the client is subscribed to.
System Groups	`system-groups`	System groups.
Activation Keys for System Groups	`system-groups-keys`	Activation keys for system groups.
Systems in System Groups	`system-groups-systems`	Clients in system groups.
System Groups Users	`system-groups-users`	System groups and users that have permissions on them.
History: System	`system-history`	Event history for each client.
History: Channels	`system-history-channels`	Channel event history.
History: Configuration	`system-history-configuration`	Configuration event history.

Report	Invoked as	Description
History: Entitlements	`system-history-entitlements`	System entitlement event history.
History: Errata	`system-history-errata`	Errata event history.
History: Kickstart	`system-history-kickstart`	Kickstart event history.
History: Packages	`system-history-packages`	Package event history.
History: SCAP	`system-history-scap`	OpenSCAP event history.
MD5 Certificates	`system-md5-certificates`	All registered clients using certificates with an MD5 checksum.
Installed Packages	`system-packages-installed`	Packages installed on clients.
System Profiles	`system-profiles`	All clients registered to the server, with software and system group information.
Users	`users`	All users registered to Uyuni.
MD5 Users	`users-md5`	All users for all organizations using MD5 encrypted passwords, with their details and roles.
Systems administered	`users-systems`	Clients that individual users can administer.

For more information about an individual report, run `spacewalk-report` with the option `--info` or `--list-fields-info` and the report name. This shows the description and list of possible fields in the report.

For further information on program invocation and options, see the `spacewalk-report(8)` man page as well as the `--help` parameter of the `spacewalk-report` command.

Tuning Changelogs

Some packages have a long list of changelog entries. This data is downloaded by default, but it is not always useful information to keep. In order to limit the amount of changelog metadata which is downloaded and to save disk space, you can put a limit on how many entries to keep on disk.

This configuration option is in the `/etc/rhn/rhn.conf` configuration file. The parameter defaults to `0`, which means unlimited.

```
java.max_changelog_entries = 0
```

If you set this parameter, it comes into effect only for new packages when they are synchronized.

After changing this parameter, restart services with `spacewalk-service restart`.

You might like to delete and regenerate the cached data to remove older data.

> **IMPORTANT**
>
> Deleting and regenerating cached data can take a long time. Depending on the number of channels you have and the amount of data to be deleted, it can potentially take several hours. The task is run in the background by Taskomatic, so you can continue to use Uyuni while the operation completes, however you should expect some performance loss.

You can delete and request a regeneration of cached data from the command line:

```
spacewalk-sql -i
```

Then on the SQL database prompt, enter:

```
DELETE FROM rhnPackageRepodata;
INSERT INTO rhnRepoRegenQueue (id, CHANNEL_LABEL, REASON, FORCE)
(SELECT sequence_nextval('rhn_repo_regen_queue_id_seq'),
        C.label,
        'cached data regeneration',
        'Y'
        FROM rhnChannel C);
\q
```

Troubleshooting

This section contains some common problems you might encounter with Uyuni, and solutions to resolving them.

Troubleshooting Corrupt Repositories

The information in the repository metadata files can become corrupt or out of date. This can create problems with updating clients. You can fix this by removing the files and regenerating it. With an new repository data file, updates should operate as expected.

Procedure: Resolving Corrupt Repository Data

1. Remove all files from `/var/cache/rhn/repodata/<channel-label>-updates-x86_64`. If you do not know the channel label, you can find it in the Uyuni {webui}, by navigating to menu:Software[Channels > Channel Label].

2. Regenerate the file from the command line:

```
spacecmd softwarechannel_regenerateyumcache <channel-label>-updates-x86_64
```

Troubleshooting Disk Space

Running out of disk space can have a severe impact on the Uyuni database and file structure which, in most cases, is not recoverable. Uyuni monitors free space in specific directories, and has configurable alerts. For more on space management, see **[Administration › Space-management ›]**.

You can recover disk space by removing unused software channels. For instructions on how to delete vendor channels, see **[Administration › Channel-management ›]**. For instructions on how to delete custom channels, see **[Administration › Custom-channels ›]**.

You can also recover disk space by cleaning up unused activation keys, content lifecycle projects, and client registrations. You can also remove redundant database entries:

Procedure: Resolving Redundant Database Entries

1. Use the `spacewalk-data-fsck` command to list any redundant database entries.

2. Use the `spacewalk-data-fsck --remove` command to delete them.

Troubleshooting Firewalls

If you are using a firewall that blocks outgoing traffic, it can either `REJECT` or `DROP` network requests. If it is set to `DROP` then you might find that synchronizing with the {scc} times out.

This occurs because the synchronization process needs to access third-party repositories that provide packages for non-{suse} clients, and not just the {scc}. When the Uyuni Server attempts to reach these repositories to check that they are valid, the firewall drops the requests, and the synchronization continues to wait for the response until it times out.

If this occurs, the synchronization takes a long time before it fails, and your non-{suse} products are not shown in the product list.

You can fix this problem in several different ways.

The simplest method is to configure your firewall to allow access to the URLs required by non-{suse} repositories. This allows the synchronization process to reach the URLs and complete successfully.

If allowing external traffic is not possible, configure your firewall to `REJECT` requests from Uyuni instead of `DROP`. This rejects requests to third-party URLs, so that the synchronization fails early rather than times out, and the products are not shown in the list.

If you do not have configuration access to the firewall, you can consider setting up a separate firewall on the Uyuni Server instead.

Troubleshooting Inactive clients

A Taskomatic job periodically pings clients to ensure they are connected. Clients are considered inactive if they have not responded to a Taskomatic check in for 24 hours or more. To see a list of inactive clients in the {webui}, navigate to menu:Systems[System List > Inactive].

Clients can become inactive for a number of reasons:

- The client is not entitled to any Uyuni service. If the client remains unentitled for 180 days (6 months), it is removed.

- On traditional clients, the `rhnsd` service has been disabled.

- The client is behind a firewall that does not allow HTTPS connections.

- The client is behind a proxy that is misconfigured.

- The client is communicating with a different Uyuni Server, or the connection has been misconfigured.

- The client is not in a network that can communicate with the Uyuni Server.

- A firewall is blocking traffic between the client and the Uyuni Server.

- Taskomatic is misconfigured.

For more information about client connections to the server, see **[Client-configuration › Contact-methods-intro ›]**.

For more information about configuring ports, see **[Installation › Ports ›]**.

For more information about troubleshooting firewalls, see **[Administration › Tshoot-firewalls ›]**.

Troubleshooting Inter-Server Synchronization

Inter-server synchronization uses caches to manage the ISS Master and Slaves. These caches can be prone to bugs that create invalid entries. In this case, bugs can show up even after updating to a version that resolves the bug, because the cache is still using invalid entries. If you upgrade to a new version of ISS, but are still experiencing problems, clear all the caches to ensure you no longer have old entries creating a problem.

Cache errors can lead to synchronization failing with a variety of errors, but the error message will usually report something like this:

```
consider removing satellite-sync cache at /var/cache/rhn/satsync/* and re-run satellite-sync
with same options.
```

You can resolve this by deleting the cache on the ISS Master and the ISS Slave, so that synchronization completes successfully.

Procedure: Resolving ISS Caching Errors

1. On the ISS Master, at the command prompt, as root, delete the cache file for the Master:

```
rm -rf /var/cache/rhn/xml-*
```

2. Restart the service:

```
rcapache2 restart
```

3. On the ISS Master, at the command prompt, as root, delete the cache file for the Slave:

```
rm -rf /var/cache/rhn/satsync/*
```

4. Restart the service:

```
rcapache2 restart
```

Troubleshooting Local Issuer Certificates

Some older bootstrap scripts create a link to the local certificate in the wrong place. This results in zypper returning an `Unrecognized error` about the local issuer certificate. You can ensure that the link to the local issuer certificate has been created correctly by checking the `/etc/ssl/certs/` directory. If you come across this problem, you should consider updating your bootstrap scripts to ensure that zypper operates as expected.

Troubleshooting Login Timeouts

By default, the Uyuni {webui} requires users to log in again after 30 minutes. Depending on your environment, you might want to adjust the login timeout value.

To adjust the value, you need to make the change in both `rhn.conf` and `web.xml`. Ensure you set the value in seconds in `/etc/rhn/rhn.conf`, and in minutes in `web.xml`. The two values must equal the same amount of time.

For example, to change the timeout value to one hour, set the value in `rhn.conf` to 3600 seconds, and the value in `web.xml` to 60 minutes.

Procedure: Adjusting the {webui} Login Timeout Value

1. Stop services:

```
spacewalk-service stop
```

2. Open `/etc/rhn/rhn.conf` and add or edit this line to include the new timeout value in seconds:

```
web.session_database_lifetime = <Timeout_Value_in_Seconds>
```

3. Save and close the file.

4. Open `/srv/tomcat/webapps/rhn/WEB-INF/web.xml` and add or edit this line to include the new timeout value in minutes:

```
<session-timeout>Timeout_Value_in_Minutes</session-timeout>
```

5. Save and close the file.

6. Restart services:

```
spacewalk-service start
```

Troubleshooting Notifications

The default lifetime of notification messages is 30 days, after which messages are deleted from the database, regardless of read status. To change this value, add or edit this line in `/etc/rhn/rhn.conf`:

```
java.notifications_lifetime = 30
```

All notification types are enabled by default. To disable a notification type, add or edit this line in `/etc/rhn/rhn.conf`:

```
java.notifications_type_disabled = OnboardingFailed,ChannelSyncFailed,ChannelSyncFinished
```

Troubleshooting OSAD and jabberd

In some cases, the maximum number of files that jabber can open is lower than the number of connected OSAD clients.

If this occurs, OSAD clients cannot contact the SUSE Manager Server, and jabberd takes an excessive amount of time to respond on port 5222.

> **NOTE**
> This fix is only required if you have more than 8192 clients connected using OSAD. In this case, we recommend you consider using Salt clients instead. For more information about tuning large scale installations, see **[Salt › Large-scale ›]**.

You can increase the number of files available to jabber by editing the jabberd local configuration file. By

default, the file is located at `/etc/systemd/system/jabberd.service.d/override.conf`.

Procedure: Adjusting the Maximum File Count

1. At the command prompt, as root, open the local configuration file for editing:

   ```
   systemctl edit jabberd
   ```

2. Add or edit this section:

   ```
   [Service]
   LimitNOFILE=<soft_limit>:<hard_limit>
   ```

   The value you choose varies depending on your environment. For example, if you have 9500 clients, increase the soft value by 100 to 9600, and the hard value by 1000 to 10500:

   ```
   [Unit]
   LimitNOFILE=
   LimitNOFILE=9600:10500
   ```

3. Save the file and exit the editor.

   > **NOTE** The default editor for systemctl files is vim. To save the file and exit, press kbd:[Esc] to enter `normal` mode, type kbd:[:wq] and press kbd:[Enter].

Ensure you also update the `max_fds` parameter in `/etc/jabberd/c2s.xml`. For example: `<max_fds>10500</max_fds>`

The soft file limit is the maximum number of open files for a single process. In Uyuni the highest consuming process is `c2s`, which opens a connection per client. 100 additional files are added, here, to accommodate for any non-connection file that `c2s` requires to work correctly. The hard limit applies to all processes belonging to jabber, and also accounts for open files from the router, `c2s` and `sm` processes.

Troubleshooting Package Inconsistencies

When packages on a client are locked, Uyuni Server may not be able to correctly determine the set of applicable patches. When this occurs, package updates are available in the {webui}, but do not appear on the client, and attempts to update the client fail. Check package locks and exclude lists to determine if packages are locked or excluded on the client.

On the client, check package locks and exclude lists to determine if packages are locked or excluded:

- On an Expanded Support Platform, check `/etc/yum.conf` and search for `exclude=`.

- On {sle} and {opensuse}, use the `zypper locks` command.

Troubleshooting Registering Cloned Clients

If you are using Uyuni to manage virtual machines, you might find it useful to create clones of your VMs. A clone is a VM that uses a primary disk that is an exact copy of an existing disk.

While cloning VMs can save you a lot of time, the duplicated identifying information on the disk can sometimes cause problems.

If you have a client that is already registered, you create a clone of that client, and then try and register the clone, you probably want Uyuni to register them as two separate clients. However, if the machine ID in both the original client and the clone is the same, Uyuni registers both clients as one system, and the existing client data is overwritten with that of the clone.

This can be resolved by changing the machine ID of the clone, so that Uyuni recognizes them as two different clients.

> **NOTE** Each step of this procedure is performed on the cloned client. This procedure does not manipulate the original client, which remains registered to Uyuni.

Procedure: Resolving Duplicate Machine IDs in Cloned Salt Clients

1. On the cloned machine, change the hostname and IP addresses. Make sure `/etc/hosts` contains the changes you made and the correct host entries.

2. For distributions that support systemd: If your machines have the same machine ID, delete the file on each duplicated client and re-create it:

   ```
   # rm /etc/machine-id
   # rm /var/lib/dbus/machine-id
   # dbus-uuidgen --ensure
   # systemd-machine-id-setup
   ```

3. For distributions that do not support systemd: Generate a machine ID from dbus:

   ```
   # rm /var/lib/dbus/machine-id
   # dbus-uuidgen --ensure
   ```

4. If your clients still have the same Salt client ID, delete the `minion_id` file on each client (FQDN is used when it is regenerated on client restart):

   ```
   # rm /etc/salt/minion_id
   ```

5. Delete accepted keys from the onboarding page and the system profile from Uyuni, and restart the client with:

   ```
   # service salt-minion restart
   ```

6. Re-register the clients. Each client now has a different `/etc/machine-id` and should be correctly displayed on the `System Overview` page.

Procedure: Resolving Duplicate Machine IDs in Cloned Traditional Clients

1. On the cloned machine, change the hostname and IP addresses. Make sure `/etc/hosts` contains the changes you made and the correct host entries.

2. Stop the `rhnsd` daemon, on {rhnminrelease6} and {sle} 11 with:

```
# /etc/init.d/rhnsd stop
```

   or, on newer systemd-based systems, with:

```
# service rhnsd stop
```

3. Stop `osad` with:

```
# /etc/init.d/osad stop
```

   or:

```
# service osad stop
```

   or:

```
# rcosad stop
```

4. Remove the `osad` authentication configuration file and the system ID:

```
# rm -f /etc/sysconfig/rhn/{osad-auth.conf,systemid}
```

5. Delete the files containing the machine IDs:

   ◦ SLES 12:

```
# rm /etc/machine-id
# rm /var/lib/dbus/machine-id
# dbus-uuidgen --ensure
# systemd-machine-id-setup
```

   ◦ SLES 11:

```
# suse_register -E
```

6. Remove the credential files:

   ◦ SLES clients:

   ```
   # rm -f /etc/zypp/credentials.d/{SCCcredentials,NCCcredentials}
   ```

   ◦ {rhel} clients:

   ```
   # rm  -f /etc/NCCcredentials
   ```

7. Re-run the bootstrap script. You should now see the cloned system in Uyuni without overriding the system it was cloned from.

Troubleshooting Registering Traditional Red Hat Clients

Traditional {rhel} 7 and SLES {es} 7 clients require some unsigned packages to work properly with Uyuni, so custom channels for these client types usually unset the `gpgcheck` flag. However, the `rhnplugin.conf` file overrides this setting, and enables the GPG check.

This means that when traditional {rhel} 7 and SLES {es} 7 clients are registered, the client will not install unsigned packages from the custom channel, even if the `gpgcheck` is disabled in the custom channel.

To resolve the problem, edit the `/etc/yum/pluginconf.d/rhnplugin.conf` file and disable the GPG check to enable installation of unsigned packages, and allow the clients to work as expected.

Troubleshooting Renaming Uyuni Server

If you change the hostname of the Uyuni Server locally, your Uyuni installation ceases to work properly. This is because the changes have not been made in the database, which prevents the changes from propagating out your clients and any proxies.

If you need to change the hostname of the Uyuni Server, you can do so using the `spacewalk-hostname-rename` script. This script updates the settings in the PostgreSQL database and the internal structures of Uyuni.

The `spacewalk-hostname-rename` script is part of the `spacewalk-utils` package.

The only mandatory parameter for the script is the newly configured IP address of the Uyuni Server.

Procedure: Renaming Uyuni Server

1. Change the network settings of the server on the system level locally and remotely at the DNS server. You also need to provide configuration settings for reverse name resolution. Changing network

settings is done in the same way as with renaming any other system.

2. Reboot the Uyuni Server to use the new network configuration and to ensure the hostname has changed.

3. Run the script `spacewalk-hostname-rename` script with the public IP address of the server. If the server is not using the new hostname, the script fails. Be aware that this script refreshes the pillar data for all Salt minion: the time it takes to run depends on the number of registered Salt systems.

4. Re-configure your clients to make your environment aware of the new hostname and IP address.

   In the Salt minion configuration file `/etc/salt/minion`, you must make sure to specify the name of the new Salt master (Uyuni Server):

   ```
   master: <new_hostname>
   ```

Once this is changed, restart the `salt-minion` process:

```
systemctl restart salt-minion
```

In order to fully propagate the hostname to the minion configuration (in the repositories URL for instance) apply the high state.

+ Traditional clients have the `/etc/sysconfig/rhn/up2date` configuration file that must be changed. With a re-activation key you can re-register traditional clients (if there are any). For more information, see [**Client-configuration** › **Registration-cli** ›]. . OPTIONAL: If you use PXE boot through a proxy, you must check the configuration settings of the proxy. On the proxy, run the `configure-tftpsync.sh` setup script and enter the requested information. For more information, see [**Installation** › **Proxy-setup** ›].

Troubleshooting RPC Connection Timeouts

RPC connections can sometimes time out due to slow networks or a network link going down. This results in package downloads or batch jobs hanging or taking longer than expected. You can adjust the maximum time that an RPC connection can take by editing the configuration file. While this does not resolve networking problems, it does cause a process to fail rather than hang.

Procedure: Resolving RPC connection timeouts

1. On the Uyuni Server, open the `/etc/rhn/rhn.conf` file and set a maximum timeout value (in seconds):

   ```
   server.timeout =`number`
   ```

2. On the Uyuni Proxy, open the `/etc/rhn/rhn.conf` file and set a maximum timeout value (in seconds):

```
proxy.timeout ='number'
```

3. On a {sles} client that uses zypper, open the `/etc/zypp/zypp.conf` file and set a maximum timeout value (in seconds):

```
## Valid values:  [0,3600]
## Default value: 180
download.transfer_timeout = 180
```

4. On a {rhel} client that uses yum, open the `/etc/yum.conf` file and set a maximum timeout value (in seconds):

```
timeout ='number'
```

NOTE   If you limit RPC timeouts to less than `180` seconds, you risk aborting perfectly normal operations.

Troubleshooting the Saltboot Formula

Because of a problem in the computed partition size value, the saltboot formula can sometimes fail when it is created on SLE 11 SP3 clients, with an error like this:

```
        ID: disk1_partitioned
  Function: saltboot.partitioned
      Name: disk1
    Result: false
   Comment: An exception occurred in this state: Traceback (most recent call last):
 File "/usr/lib/python2.6/site-packages/salt/state.py", line 1767, in call
   **cdata['kwargs'])
 File "/usr/lib/python2.6/site-packages/salt/loader.py", line 1705, in wrapper
   return f(*args, **kwargs)
 File "/var/cache/salt/minion/extmods/states/saltboot.py", line 393, in disk_partitioned
   existing = __salt__['partition.list'](device, unit='MiB')
 File "/usr/lib/python2.6/site-packages/salt/modules/parted.py", line 177, in list_
   'Problem encountered while parsing output from parted')
CommandExecutionError: Problem encountered while parsing output from parted
```

This problem can be resolved by manually configuring the size of the partition containing the operating system. When the size is set correctly, formula creation works as expected.

Procedure: Manually Configuring the Partition Size in the Saltboot Formula

1. In the Uyuni {webui}, navigate to menu:Systems[System Groups] and select the `Hardware Type Group` that contains the SLE 11 SP3 client that is causing the error. In the `Formulas` tab, navigate to the `Saltboot` subtab.

2. Locate the partition that contains the operating system, and in the `Partition Size` field, type the appropriate size (in MiB).

3. Click btn:[Save Formula], and apply the highstate to save your changes.

Troubleshooting Synchronization

Synchronization can fail for a number of reasons. To get more information about a connection problem, run this command:

```
export URLGRABBER_DEBUG=DEBUG
spacewalk-repo-sync -c <channelname> <options> > /var/log/spacewalk-repo-sync-$(date +%F-%R).log 2>&1
```

You can also check logs created by Zypper at `/var/log/zypper.log`

GPG Key Mismatch

Uyuni does not automatically trust third party GPG keys. If package synchronization fails, it could be because of an untrusted GPG key. You can find out if this is the case by opening `/var/log/rhn/reposync` and looking for an error like this:

```
['/usr/bin/spacewalk-repo-sync', '--channel', 'sle-12-sp1-ga-desktop-
nvidia-driver-x86_64', '--type', 'yum', '--non-interactive']
RepoMDError: Cannot access repository. Maybe repository GPG keys are not imported
```

To resolve the problem, you need to import the GPG key to Uyuni. For more on importing GPG keys, see **[Administration › Repo-metadata ›]**.

Checksum Mismatch

If a checksum has failed, you might see an error like this in the `/var/log/rhn/reposync/*.log` log file:

```
Repo Sync Errors: (50, u'checksums did not match
326a904c2fbd7a0e20033c87fc84ebba6b24d937 vs
afd8c60d7908b2b0e2d95ad0b333920aea9892eb', 'Invalid information uploaded
to the server')
The package microcode_ctl-1.17-102.57.62.1.x86_64 which is referenced by
patch microcode_ctl-8413 was not found in the database. This patch has
been skipped.
```

You can resolve this error by running the synchronization from the command prompt with the `-Y` option:

```
spacewalk-repo-sync --channel <channelname> -Y
```

This option verifies the repository data before the synchronization, rather than relying on locally cached checksums.

Connection Timeout

If the download times out with the following error:

```
28, 'Operation too slow. Less than 1000 bytes/sec transferred the last 300 seconds
```

You can resolve this error by specifying `reposync_timeout` and `reposync_minrate` configuration values in `/etc/rhn/rhn.conf`. By default, when less than 1000 bytes per second are transferred in 300 secs, the download is aborted. You can adjust the number of bytes per second with `reposync_minrate`, and the number of seconds to wait with `reposync_timeout`.

Troubleshooting Taskomatic

Repository metadata regeneration is a relatively intensive process, so Taskomatic can take several minutes to complete. Additionally, if Taskomatic crashes, repository metadata regeneration can be interrupted.

If Taskomatic is still running, or if the process has crashed, package updates can seem available in the {webui}, but do not appear on the client, and attempts to update the client fail. In this case, the `zypper ref` command shows an error like this:

```
Valid metadata not found at specified URL
```

To correct this, determine if Taskomatic is still in the process of generating repository metadata, or if a crash could have occurred. Wait for metadata regeneration to complete or restart Taskomatic after a crash in order for client updates to be carried out correctly.

Procedure: Resolving Taskomatic Problems

1. On the Uyuni Server, check the `/var/log/rhn/rhn_taskomatic_daemon.log` file to determine if any metadata regeneration processes are still running, or if a crash occurred.

2. Restart taskomatic:

   ```
   service taskomatic restart
   ```

In the Taskomatic log files, you can identify the section related to metadata regeneration by looking for opening and closing lines that look like this:

```
<YYYY-DD-MM> <HH:MM:SS>,174 [Thread-584] INFO
com.redhat.rhn.taskomatic.task.repomd.RepositoryWriter - Generating new repository metadata
for channel 'cloned-2018-q1-sles12-sp3-updates-x86_64'(sha256) 550 packages, 140 errata

...

<YYYY-DD-MM> <HH:MM:SS>,704 [Thread-584] INFO
com.redhat.rhn.taskomatic.task.repomd.RepositoryWriter - Repository metadata generation for
'cloned-2018-q1-sles12-sp3-updates-x86_64' finished in 4 seconds
```

GNU Free Documentation License

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections

then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
    Permission is granted to copy, distribute and/or modify this document
    under the terms of the GNU Free Documentation License, Version 1.2
    or any later version published by the Free Software Foundation;
    with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
    A copy of the license is included in the section entitled{ldquo}GNU
    Free Documentation License{rdquo}.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the " with…Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.