Administration Guide

Uyuni 4.0

March 16, 2019

Table of Contents

Unresolved directive in nav-administration-guide.adoc - include::/modules/administration/pages/common_gfdl1.2_i.adoc[leveloffset=+1]

Introduction

TODO: Introduction to the Admin Guide

Image Building and Management

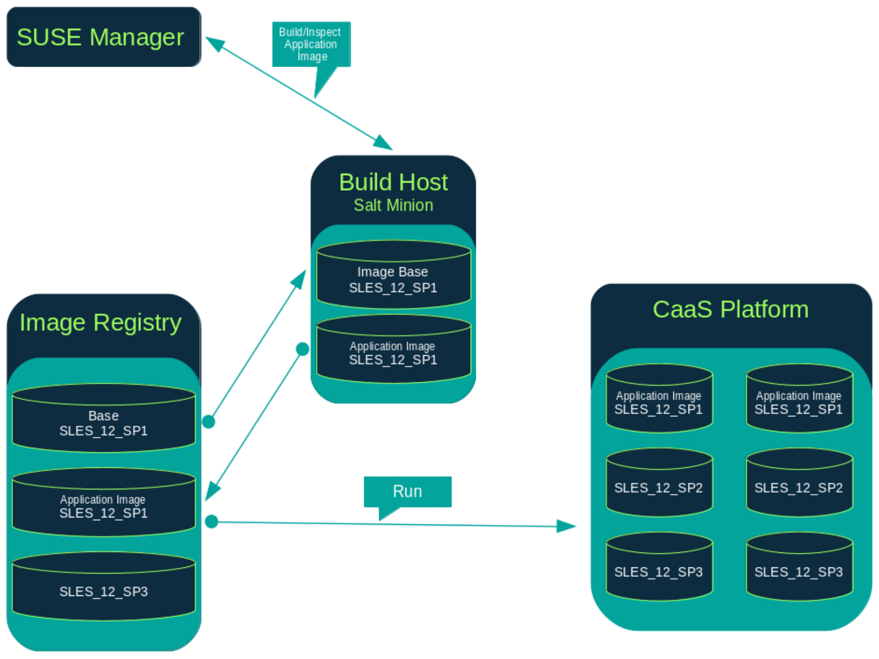
Image Building Overview

Uyuni enables system administrators to build containers, systems, and virtual images. Uyuni helps with creating Image Stores and managing Image Profiles.

Uyuni supports two distinct build types:

- Dockerfile-for more information, see Container Images
- Kiwi image system-for more information, see OS Images

Container Images



Requirements

The containers feature is available for Salt minions running SUSE Linux Enterprise Server 12 or later. Before you begin, ensure your environment meets these requirements:

- An existing external GitHub or internal GitLab repository containing a Dockerfile and configuration scripts (example scripts are provided in this chapter).
- A properly configured image registry.

Registry Provider Solutions

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 CaaS Platform, see:

- SUSE Linux Enterprise Server 12 SP3 Docker Guide
- SUSE CaaS Platform 2 Documentation

Creating a Build Host

To build images with Uyuni, you will need to create and configure a build host. Container build hosts are Salt minions running SUSE Linux Enterprise 12 or later. This section guides you though the initial configuration for a build host.

From the Uyuni Web UI perform these steps to configure a build host.

1. Select a minion to be designated as a build host from the **Systems** › **Overview** page.
2. From the `System Details` page for the selected minion assign the containers modules by going to **Software** › **Software Channels** and enabling `SLE-Module-Containers12-Pool` and `SLE-Module-Containers12-Updates`. Confirm by clicking **[Change Subscriptions]**.
3. From the **System Details** › **Properties** page, enable `Add-on System Type` and `Container Build Host` and confirm by clicking **[Update Properties]**.
4. Install all required packages by applying `Highstate`. From the system details page select **States** › **Highstate** and click `Apply Highstate`. Alternatively, apply Highstate from the Uyuni Server command line:

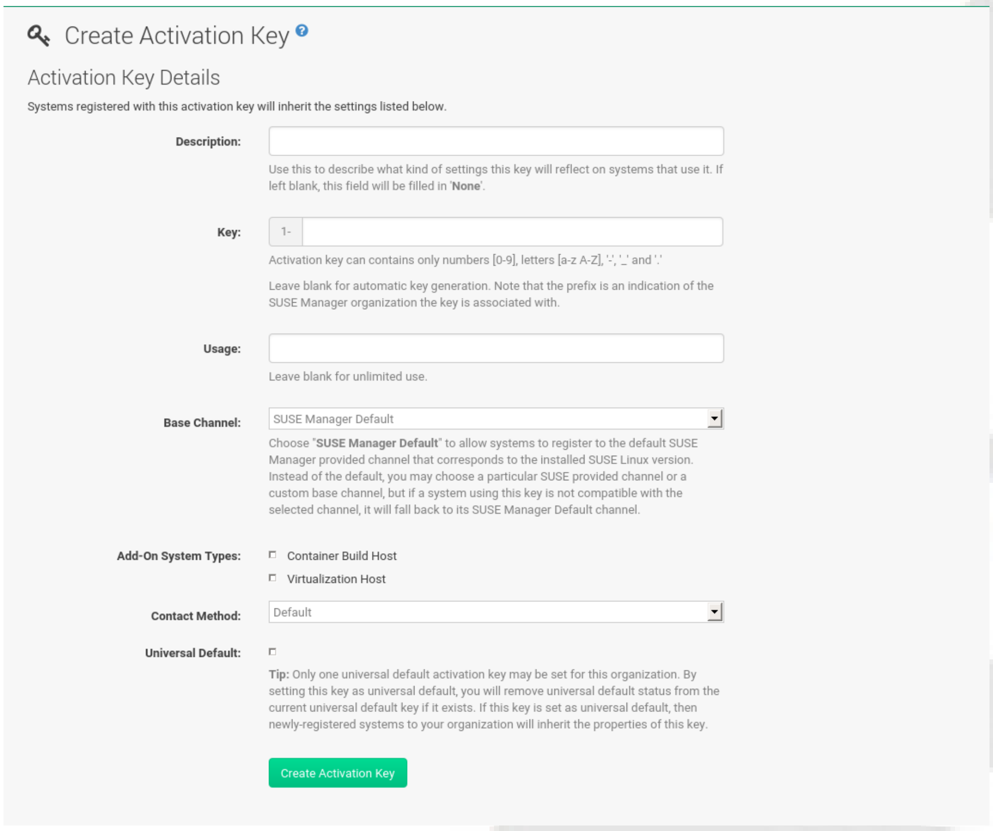
```
salt '$your_minion' state.highstate
```

Define Container Build Channels with an Activation Key

Create an activation key associated with the channel that your images will use.

Relationship Between Activation Keys and Image Profiles

To build containers, you will need an activation key that is associated with a channel other than "SUSE Manager Default".


Create Activation Key

Activation Key Details

Systems registered with this activation key will inherit the settings listed below.

Description: Use this to describe what kind of settings this key will reflect on systems that use it. If left blank, this field will be filled in 'None'.

Key: 1- Activation key can contains only numbers [0-9], letters [a-z A-Z], '-', '_' and '.'

Leave blank for automatic key generation. Note that the prefix is an indication of the SUSE Manager organization the key is associated with.

Usage: Leave blank for unlimited use.

Base Channel: SUSE Manager Default

Choose "SUSE Manager Default" to allow systems to register to the default SUSE Manager provided channel that corresponds to the installed SUSE Linux version. Instead of the default, you may choose a particular SUSE provided channel or a custom base channel, but if a system using this key is not compatible with the selected channel, it will fall back to its SUSE Manager Default channel.

Add-On System Types: Container Build Host, Virtualization Host

Contact Method: Default

Universal Default:

Tip: Only one universal default activation key may be set for this organization. By setting this key as universal default, you will remove universal default status from the current universal default key if it exists. If this key is set as universal default, then newly-registered systems to your organization will inherit the properties of this key.

1. Select **Main Menu › Systems › Activation Keys**.
2. Click **[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 **[Create Activation Key]**.

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

Creating an Image Store

Define a location to store all of your images by creating an Image Store.

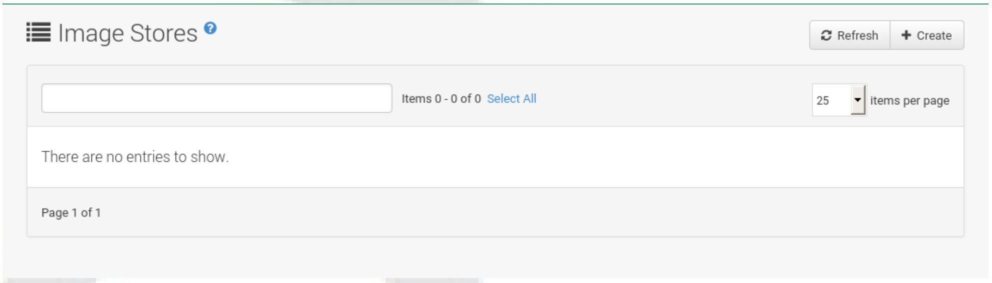
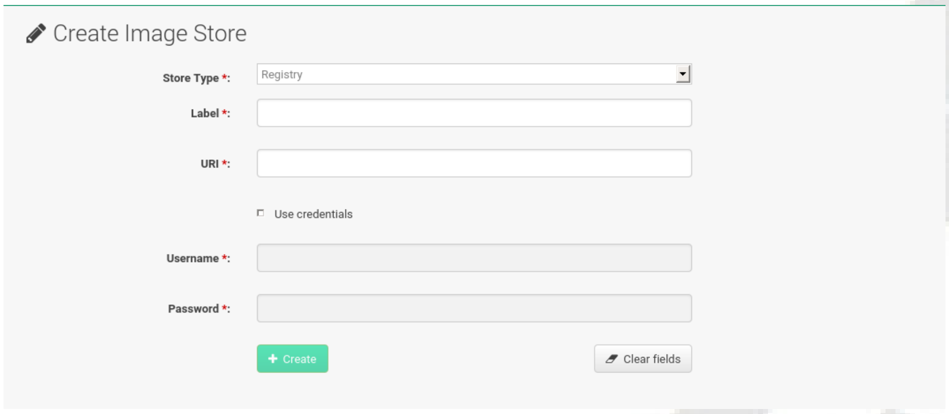


Image Stores

There are no entries to show.

Page 1 of 1

1. Select **Main Menu › Images › Stores**.
2. Click `Create` to create a new store.



3. Uyuni currently provides support only for the `Registry` store type. 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
   ```

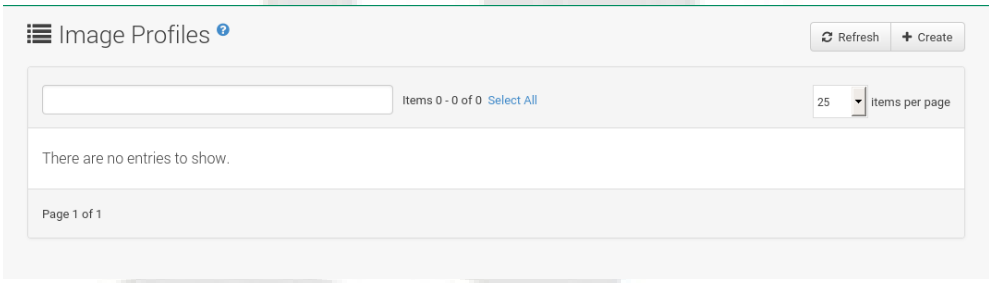
   The Registry URI can also be used to specify an image store on a used registry.

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

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

Creating an Image Profile

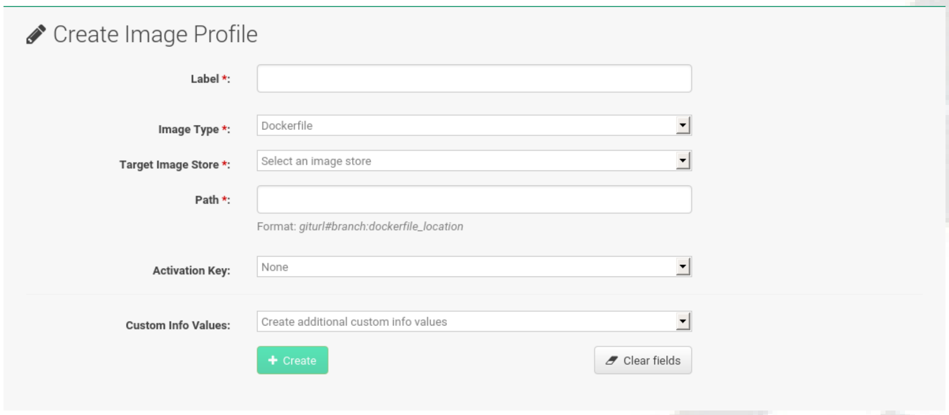
Manage Image Profiles from the `Image Profile` page.



There are no entries to show.

Procedure: Create an Image Profile

1. To create an image profile select **Image › Profiles** and click **[Create]**.


Create Image Profile

Label *:	
Image Type *:	Dockerfile
Target Image Store *:	Select an image store
Path *:	Format: giturl#branch:dockerfile_location
Activation Key:	None
Custom Info Values:	Create additional custom info values

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

Only lowercase characters are permitted in container labels. 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. Enter a Github or Gitlab repository URL (http, https, or token authentication) in the `Path` field using one of the following 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 and not publicly accessible, you need to modify the profile's git 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/
```

Specifying a Github or Gitlab Branch

If a branch is not specified, the `master` branch will be 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.

Relationship Between Activation Keys and Image Profiles

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

2. Click the **[Create]** button.

Example Dockerfile and add_packages Script

This section contains an example Dockerfile. You specify a Dockerfile that will be used during image building when creating an image profile. A Dockerfile and any associated scripts should be stored within an internal or external Github or Gitlab repository:

Required Dockerfile Lines

The Dockerfile provides access to a specific repository version served by Uyuni. This example Dockerfile is used by Uyuni to trigger a build job on a build host minion. The `ARG` parameters ensure that the image that is built is associated with the desired repository version served by Uyuni. The `ARG` parameters also allow you to build image versions of SUSE Linux Enterprise Server which may differ from the version of SUSE Linux Enterprise Server 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 version 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
```

This is an example `add_packages.sh` script for use with your Dockerfile:

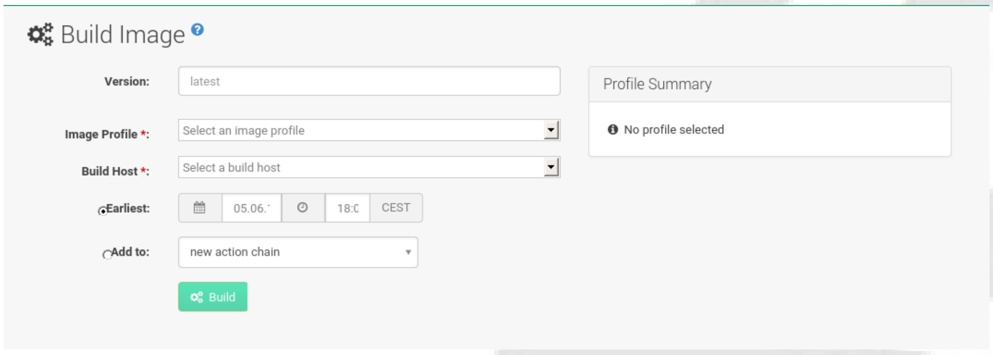
```
#!/bin/bash
set -e

zypper --non-interactive --gpg-auto-import-keys ref

zypper --non-interactive in python python-xml aaa_base aaa_base-extras net-tools timezone vim
less sudo tar
```

Packages Required for Inspecting Your Images

To inspect images and provide the package and product list of a container to the Uyuni Web UI you will need to install python and python-xml within the container. If these packages remain uninstalled, your images will still build, but the package and product list will be unavailable from the Web UI.

Building an Image

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



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

Profile Summary

No profile selected

Procedure: Build an Image

1. For this example select **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`.

Profile Summary

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

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

Importing an Image

You can import and inspect arbitrary images. Select **Images › Images** from the left navigation bar. Complete the text boxes of the `Import` dialog. Once it has processed, the imported image will be listed on the `Images` page.

Procedure: Import an Image

1. From **Images › Images** click **[Import]** to open the `Import Image` dialog.
2. In the `Import Image` dialog complete these fields:

Image store
: The registry from where the image will be 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 will pull and inspect the image.

Activation key
: The activation key that provides the path to the software channel that the image will be inspected with.

For confirmation, click **[Import]**.

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

Once it has been processed, you can find the imported image in the `Images` list. It has a different icon in the `Build` column, to indicate that the image is imported (see screenshot). The status icon for the imported image can also be seen on the `Overview` tab for the image.

Troubleshooting

These are some known problems that you might encounter when working with images:

- HTTPS certificates to access the registry or the git repositories should be deployed to the minion by a custom state file.
- SSH git access using Docker is currently unsupported. You may test it, but SUSE will not provide support.
- If the python and python-xml packages are not installed in your images during the build process, Salt cannot run within the container and reporting of installed packages or products will fail. This will result in an `unknown` update status.

OS Images

OS images are built by the Kiwi image system. They can be of various types: PXE, QCOW2, LiveCD images, and others.

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

Requirements

The Kiwi image building feature is available for Salt minions running SUSE Linux Enterprise Server 12. It is currently not supported to build SUSE Linux Enterprise 15 images.

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

- Git repository
- HTTP hosted tarball
- Local build host directory

Example scripts are provided in the following sections.

Hardware Requirements for Hosts Running OS Images

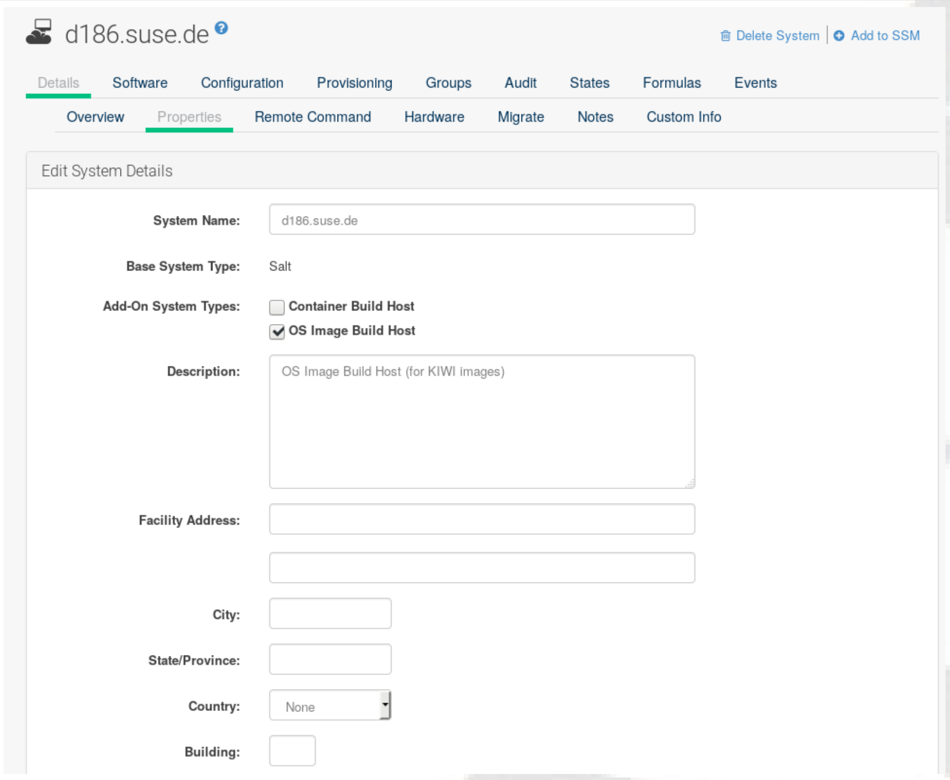
Hosts running OS images built with Kiwi need at least 1 GB of RAM. Disk space depends on the actual size of the image. For more information, see the documentation of the underlying system.

Creating a Build Host

To build all kinds of images with Uyuni, create and configure a build host. OS image build hosts are Salt minions running SUSE Linux Enterprise Server 12 (SP3 or later). This procedure will guide you though the initial configuration for a build host.

From the Uyuni Web UI perform these steps to configure a build host:

1. Select a minion that will be designated as a build host from the **Main Menu** › **Systems** › **Overview** page.
2. From the **System Details** › **Properties** page, enable the `Add-on System Type: OS Image Build Host` and confirm with **[Update Properties]**.


d186.suse.de

Delete System | Add to SSM

Details | Software | Configuration | Provisioning | Groups | Audit | States | Formulas | Events

Overview | Properties | Remote Command | Hardware | Migrate | Notes | Custom Info

Edit System Details

System Name:	d186.suse.de
Base System Type:	Salt
Add-On System Types:	☐ Container Build Host ☑ OS Image Build Host
Description:	OS Image Build Host (for KIWI images)
Facility Address:	
City:	
State/Province:	
Country:	None
Building:	

3. From the **System Details › Software › Software Channels** page, enable `SLE-Manager-Tools12-Pool` and `SLE-Manager-Tools12-Updates` (or a later version). Schedule and click **[Confirm]**.
4. Install Kiwi and all required packages by applying Highstate. From the system details page select **States › Highstate** and click **[Apply Highstate]**. Alternatively, apply Highstate from the Uyuni Server command line:

```
salt '$your_minion' 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 will call the package script using the default values. However if the certificate path was changed or unavailable, you will need to call the package script manually using `--ca-cert-full-path <path_to_certificate>` after the upgrade procedure has finished.

Listing 1. 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`.

The RPM Package with the Uyuni Certificate Must Be Specified in the Build Source

Make sure your build source Kiwi configuration contains `rhn-org-trusted-ssl-cert-osimage` as a required package in the `bootstrap` section.

Listing 2. config.xml

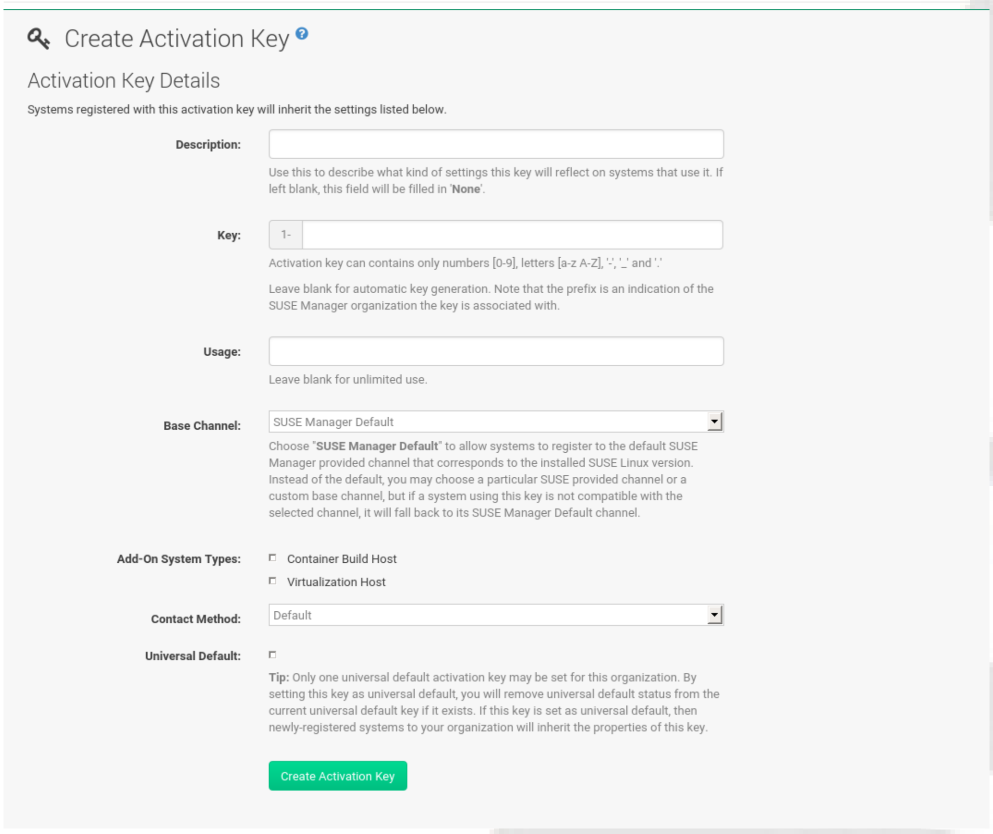
```
...
  <packages type="bootstrap">
    ...
    <package name="rhn-org-trusted-ssl-cert-osimage"
bootinclude="true"/>
  </packages>
...
```

Define Kiwi Build Channels with an Activation Key

Create an activation key associated with the channel that your images will use. Activation keys are mandatory for OS Image building.

Relationship Between Activation Keys and Image Profiles

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



Create Activation Key

Activation Key Details

Systems registered with this activation key will inherit the settings listed below.

Description:	Use this to describe what kind of settings this key will reflect on systems that use it. If left blank, this field will be filled in 'None'.
Key:	1- Activation key can contains only numbers [0-9], letters [a-z A-Z], '-', '_' and '.' Leave blank for automatic key generation. Note that the prefix is an indication of the SUSE Manager organization the key is associated with.
Usage:	Leave blank for unlimited use.
Base Channel:	SUSE Manager Default Choose "SUSE Manager Default" to allow systems to register to the default SUSE Manager provided channel that corresponds to the installed SUSE Linux version. Instead of the default, you may choose a particular SUSE provided channel or a custom base channel, but if a system using this key is not compatible with the selected channel, it will fall back to its SUSE Manager Default channel.
Add-On System Types:	☐ Container Build Host ☐ Virtualization Host
Contact Method:	Default
Universal Default:	☐ Tip: Only one universal default activation key may be set for this organization. By setting this key as universal default, you will remove universal default status from the current universal default key if it exists. If this key is set as universal default, then newly-registered systems to your organization will inherit the properties of this key.

1. In the Web UI, select **Main 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 **[Create Activation Key]**.

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

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 will be located at `/srv/www/os-images`.

Image stores for Kiwi build type

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>`

Creating an Image Profile

Manage Image Profiles using the Web UI.

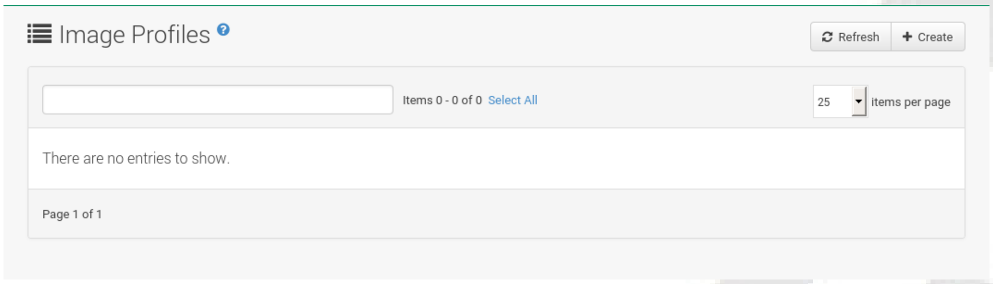


Image Profiles

Items 0 - 0 of 0 Select All

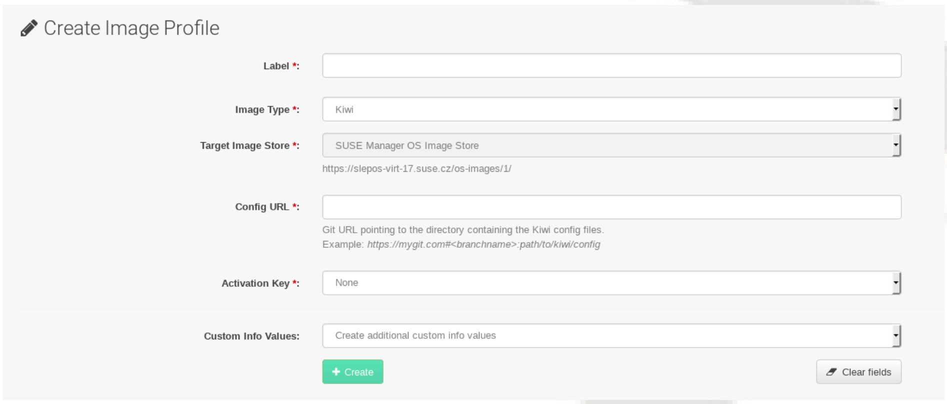
25 items per page

There are no entries to show.

Page 1 of 1

Procedure: Create an Image Profile

1. To create an image profile select from **Main Menu** › **Images** › **Images** › **Profiles** and click **[Create]**.


Create Image Profile

Label *:	
Image Type *:	Kiwi
Target Image Store *:	SUSE Manager OS Image Store https://slepos-virt-17.suse.cz/os-images/1/
Config URL *:	Git URL pointing to the directory containing the Kiwi config files. Example: *https://mygit.com#<branchname>:path/to/kiwi/config*
Activation Key *:	None
Custom Info Values:	Create additional custom info values

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

   Relationship Between Activation Keys and Image Profiles

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

7. Confirm with the **[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.

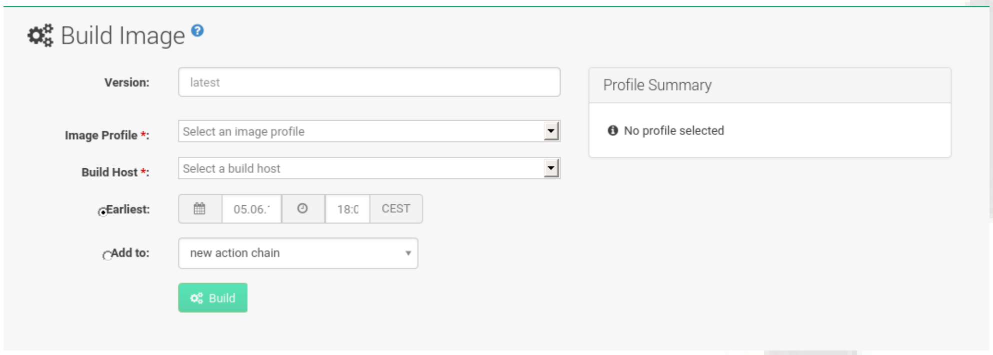
SUSE provides examples of fully functional image sources at the SUSE/manager-build-profiles public GitHub repository.

Listing 3. 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>
```

Building an Image

There are two ways to build an image using the Web UI. Either select **Main Menu › Images › Build**, or click the build icon in the **Main 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: Build an Image

1. Select **Main 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`.

   Profile Summary

   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 will show up in this area.

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

Image Inspection and Salt Integration

After the image is successfully built, the inspection phase begins. During the inspection phase SUSE Manager collects information about the image:

- List of packages installed in the image
- Checksum of the image
- Image type and other image details

If the built image type is `PXE`, a Salt pillar will also be 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 minions.

Troubleshooting

Building an image requires of several dependent steps. When the build fails, investigation of salt states results can help you to identify the source of the failure. Usual 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 Creating a Build Host.

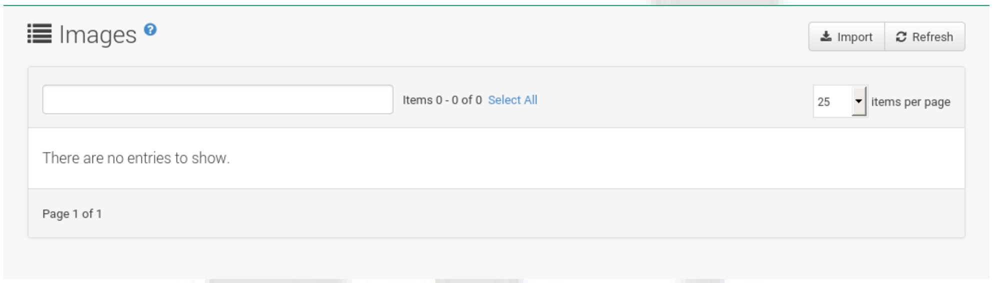
Limitations

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 minion by a custom state file, or configured manually.
- Importing Kiwi-based images is not supported.

Listing Image Profiles Available for Building

To list images available for building select **Main Menu › Images › Images**. A list of all images will be displayed.



Images

Import Refresh

Items 0 - 0 of 0 Select All

25 items per page

There are no entries to show.

Page 1 of 1

Displayed data about images includes an image `Name`, its `Version` and the build `Status`. You will also see the image update status with a listing of possible patch and package updates that are available for the image.

Clicking the **[Details]** button on an image will provide a detailed view including an exact list of relevant patches and a list of all packages installed within the image.

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

Live Patching with SUSE Manager

Introduction

Under normal circumstances a system needs to be rebooted after a kernel update. SLE Live Patching allows you skipping the reboot by applying a subset of Linux kernel releases injected via kGraft live patching technology.

In the following sections you will learn how to use SLE Live Patching to avoid the typical reboot requirement after updating a system kernel.

For in depth information covering kGraft use, see https://www.suse.com/documentation/sles-12/singlehtml/book_sle_admin/book_sle_admin.html#cha.kgraft.

Initial Setup Requirements

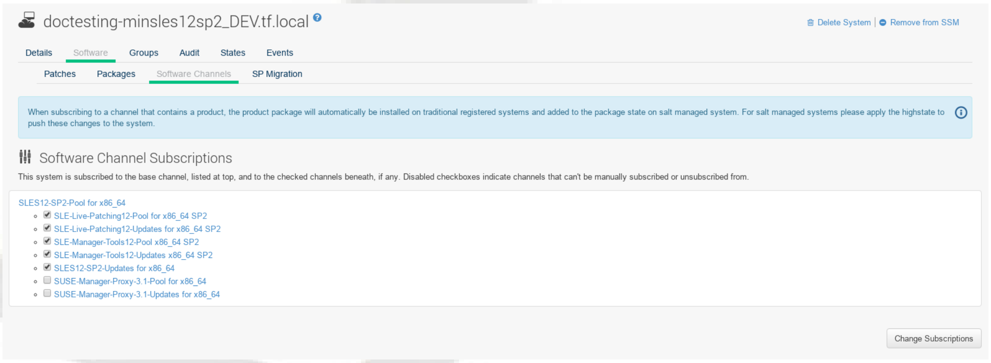
To work with SLE Live Patching the following expectations are assumed:

- Uyuni fully updated.
- At least 1 Salt Minion running SLES 12 SP1 or later and registered with Uyuni .
- The matching SLES 12 SPx channels including the SLE Live Patching child channel fully synced.

Live Patching Setup

1. Subscribe all systems to be managed via live patching to your fully synced live patching child channels within your systems base channel by browsing to **Software › Software Channels** . Select both live patching channels and change subscription.

When subscribing to a channel that contains a product, the product package will automatically be installed on traditionaly registered systems and added to the package state on Salt managed systems. For Salt managed systems please apply the highstate to push these changes to your systems.


doctesting-minsles12sp2_DEV.tf.local
Delete System | Remove from SSM
Details Software Groups Audit States Events
Patches Packages Software Channels SP Migration

When subscribing to a channel that contains a product, the product package will automatically be installed on traditional registered systems and added to the package state on salt managed system. For salt managed systems please apply the highstate to push these changes to the system.

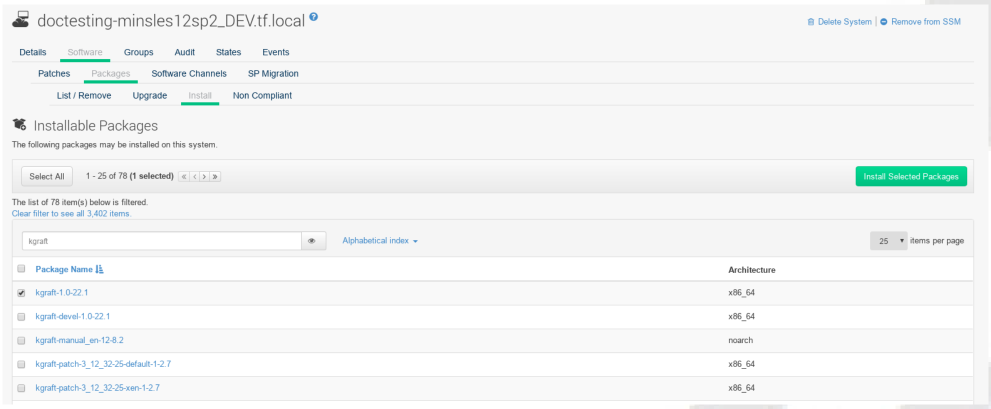
Software Channel Subscriptions

This system is subscribed to the base channel, listed at top, and to the checked channels beneath, if any. Disabled checkboxes indicate channels that can't be manually subscribed or unsubscribed from.

SLES12-SP2-Pool for x86_64
- ✓ SLE-Live-Patching12-Pool for x86_64 SP2
- ✓ SLE-Live-Patching12-Updates for x86_64 SP2
- ✓ SLE-Manager-Tools12-Pool x86_64 SP2
- ✓ SLE-Manager-Tools12-Updates x86_64 SP2
- ✓ SLES12-SP2-Updates for x86_64
- ☐ SUSE-Manager-Proxy-3.1-Pool for x86_64
- ☐ SUSE-Manager-Proxy-3.1-Updates for x86_64

1. Use the search field listed under **Software › Packages › Install** to install the latest `kgraft` package

to all systems to be managed via live patching.



doctesting-minsles12sp2_DEV.tf.local

Delete System | Remove from SSM

Details | Software | Groups | Audit | States | Events

Patches | Packages | Software Channels | SP Migration

List / Remove | Upgrade | Install | Non Compliant

Installable Packages

The following packages may be installed on this system.

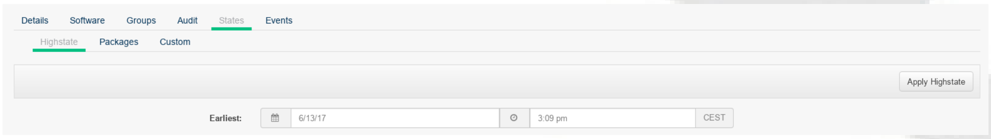
Select All   1 - 25 of 78 **(1 selected)**   Install Selected Packages

The list of 78 item(s) below is filtered.
Clear filter to see all 3,402 items.

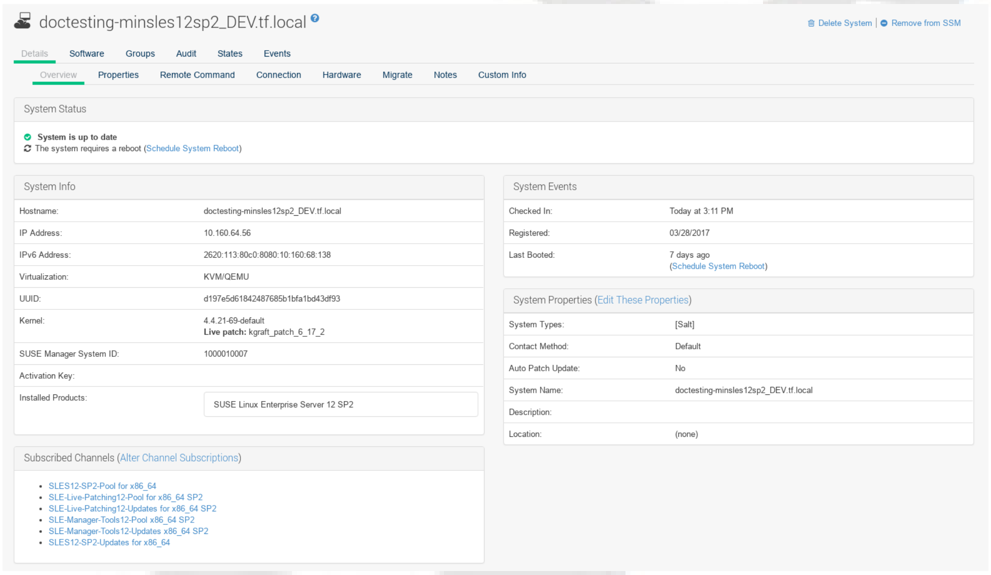
kgraft   Alphabetical index   25 items per page

	Package Name	Architecture
✓	kgraft-1.0-22.1	x86_64
	kgraft-devel-1.0-22.1	x86_64
	kgraft-manual_en-12-8.2	noarch
	kgraft-patch-3_12_32-25-default-1-2.7	x86_64
	kgraft-patch-3_12_32-25-xen-1-2.7	x86_64

1. Apply the highstate to enable live patching:



1. Once the highstate has been applied on Salt systems or the package has been installed on traditional systems browse to the systems details page for confirmation that live patching has been enabled. You can check the live patching state listed under the **System Info › Kernel** table field:



doctesting-minsles12sp2_DEV.tf.local

Delete System | Remove from SSM

Details | Software | Groups | Audit | States | Events

Overview | Properties | Remote Command | Connection | Hardware | Migrate | Notes | Custom Info

System Status

System is up to date
The system requires a reboot (Schedule System Reboot)

System Info

Hostname:	doctesting-minsles12sp2_DEV.tf.local
IP Address:	10.160.64.56
IPv6 Address:	2620:113:80c0:8080:10:160:68:138
Virtualization:	KVM/QEMU
UUID:	d197e5d61842487685b1bfa1bd43df93
Kernel:	4.4.21-69-default **Live patch:** kgraft_patch_6_17_2
SUSE Manager System ID:	1000010007
Activation Key:	
Installed Products:	SUSE Linux Enterprise Server 12 SP2

System Events

Checked In:	Today at 3:11 PM
Registered:	03/28/2017
Last Booted:	7 days ago (Schedule System Reboot)

System Properties (Edit These Properties)

System Types:	[Salt]
Contact Method:	Default
Auto Patch Update:	No
System Name:	doctesting-minsles12sp2_DEV.tf.local
Description:	
Location:	(none)

Subscribed Channels (Alter Channel Subscriptions)

- SLES12-SP2-Pool for x86_64
- SLE-Live-Patching12-Pool for x86_64 SP2
- SLE-Live-Patching12-Updates for x86_64 SP2
- SLE-Manager-Tools12-Pool x86_64 SP2
- SLE-Manager-Tools12-Updates x86_64 SP2
- SLES12-SP2-Updates for x86_64

Cloning Channels

It is considered best practice to clone a vendor channel that will be modified into a new channel with one of the following prefix names: `dev`, `testing`, and `prod`. In the following procedure you will clone the default vendor channel into a new channel named `dev-sles12-sp3-pool-x86_64` using the command line.

1. Open a terminal and as root enter:

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

 1. sles{sles-version}-sp{sp-ver}-pool-x86_64
      \__ sle-live-patching{sles-version}-pool-x86_64-sp{sp-ver}\__ sle-live-
patching{sles-version}-updates-x86_64-sp{sp-ver}\__ sle-manager-tools{sles-version}-
pool-x86_64-sp{sp-ver}\__ sle-manager-tools{sles-version}-updates-x86_64-sp{sp-ver}\__
sles{sles-version}-sp{sp-ver}-updates-x86_64
```

2. Now use the *--init* argument to automatically create a new development clone of the original vendor channel:

```
spacewalk-manage-channel-lifecycle --init -c sles{sles-version}-sp{sp-ver}-pool-x86_64
```

Removing Non-live Kernel Patches from the Cloned Channels

In the following procedure you will remove all kernel patch updates located in the dev-sles12 -spSP3 -updates-x86_64 channel that require a reboot. You created dev-sles12 -spSP3 -updates-x86_64 during [proc.live.patching.clones].

1. Check the current kernel version in use on your client:

```
# uname -r
3.12.62-60.64.8-default
```

2. From the Uyuni Web UI select **Software › Manage Software Channels › Overview › dev-sles12-sp3-updates-x86_64 › Patches › List/Remove** . Type `kernel` in the search field. Find the kernel version that matches the kernel in use on your minion.
3. Remove all kernel update versions that are later than the current kernel.
4. Your channel is now ready to promote for testing SLE Live Patching.

Promoting Channels

The following procedure will guide you through promoting and cloning a development channel to a testing channel. You will change the subscription from the dev repositories on your client to the new testing channel repositories. You will also add the SLE Live Patching child channels to your client.

1. Promote and clone the `dev-sles12 -spSP3 -pool-x86_64` to a new testing channel:

```
{prompt.root}spacewalk-manage-channel-lifecycle -promote -c dev-sles{sles-version}-
sp{sp-ver}-pool-x86_64
```

2. From the Uyuni Web UI under the **Systems ›] tab select your client system to view the**

menu:System Details[page. Select **Software › Software Channels** . From the Software Channels page you can edit which channels a system is subscribed to. Select the new base software channel, in this case it should be `test-sles12-sp3-pool-x86_64` . Click the **Confirm ›] button to switch the Base Software Channel and finalize it by clicking the menu:Modify Base Software Channel[** button.

3. From the **Software Channels ›] page select and add both SLE Live Patching child channels by clicking the menu:Change Subscriptions[** button.

Applying Live Patches to a Kernel

The following procedure will guide you through selecting and viewing available CVE Patches (Common Vulnerabilities and Exposures) then applying these kernel updates using the new SLE Live Patching feature.

1. Select your SLES 12 SPSP3 minion from the **Systems ›] page to view its menu:System Details[** . Once you have added the SLES 12 SPSP3 Updates child channel to your client, you should see several `Critical` software updates available. Click on `Critical` to see a list of available patches. Select any of these patches listed with the following synopsis: *Important: Security update for the Linux kernel*. All fixed security bugs will be listed along with their number. For example:(CVE-2016-8666)

Reboot Icon

Normal or non-live kernel patches always require a reboot. In Uyuni these are represented by a `Reboot Required` icon located next to the `Security` shield icon.

2. You can search for individual CVE’s by selecting the **Audit** tab from the navigation menu. Try searching for `CVE-2016-8666`. You will see that the patch is available in the vendor update channel and the systems it applies to will be listed.

CVE Availability

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 will required a reboot. The assigned CVE numbers for these issues are not included in live patches. A CVE audit will display this requirement.

Monitoring with Icinga

Introduction

This chapter provides guidance on the setup of an Icinga server using SLES 12 SP3. For more information, see the Official Icinga documentation: http://docs.icinga.org/latest/en/.

Installation and Basic Configuration

Icinga packages are found in the `SLE-Manager-Tools12-Updates x86_64`.

Icinga Installation Location

Do not install Icinga on the Uyuni server. Install Icinga on a stand-alone SUSE Linux Enterprise client.

Procedure: Installation and Basic Configuration

1. Register the new client with Uyuni and subscribe it to the Uyuni client and update channels. SLES 12 and later include these channels by default.
2. Install the required Icinga packages on the new client:

```
zypper in icinga icinga-idoutils-pgsql postgresql postgresql94-server \
monitoring-plugins-all apache2
```

3. Edit the `/etc/icinga/objects/contacts.cfg` file and add the email address which you will use for reciving alerts.

```
define contact {
  contact_name      icingaadmin         ; Short name of user
  use               generic-contact     ; Inherit default values
  alias             Icinga Admin        ; Full name of user
  email             icinga@localhost    ; <<*** CHANGE THIS TO YOUR EMAIL ADDRESS ***
}
```

4. Enable postgres on boot and start the database:

```
systemctl enable postgresql.service
systemctl start postgresql.service
```

5. Create the database and user for Icinga:

```
>psql
 postgres=# ALTER USER postgres WITH PASSWORD '<newpassword>';
 postgres=# CREATE USER icinga;
 postgres=# ALTER USER icinga WITH PASSWORD 'icinga';
 postgres=# CREATE DATABASE icinga;
 postgres=# GRANT ALL ON DATABASE icinga TO icinga;
 postgres=# \q
 exit
```

6. Adjust client authentication rights located in `/var/lib/pgsql/data/pg_hba.conf` to match the following:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   icinga         icinga                                  trust
local   all            postgres                                ident

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
#host    replication     postgres        127.0.0.1/32            ident
#host    replication     postgres        ::1/128                 ident
```

Placement of Authentication Settings

Ensure the local entries for icinga authentication settings are placed above all other local entries or you will get an error when configuring the database schema. The entries in `pg_hba.conf` are read from top to bottom.

7. Reload the Postgres service:

```
systemctl reload postgresql.service
```

8. Configure the database schema by running the following command in `/usr/share/doc/packages/icinga-idoutils-pgsql/pgsql/`:

```
psql -U icinga -d icinga < pgsql.sql
```

9. Edit the following lines in `/etc/icinga/ido2db.cfg` to switch from the default setting of mysql to postgres:

```
vi /etc/icinga/ido2db.cfg

 db_servertype=pgsql
 db_port=5432
```

Open Firewall Port

Allow port `5432` through your firewall or you will not be able to access the WebGUI.

10. Create an icinga admin account for logging into the web interface:

    ```
    htpasswd -c /etc/icinga/htpasswd.users icingaadmin
    ```

11. Enable and start all required services:

    ```
    systemctl enable icinga.service
    systemctl start icinga.service
    systemctl enable ido2db.service
    systemctl start ido2db.service
    systemctl enable apache2.service
    systemctl start apache2.service
    ```

12. Login to the WebGUI at: http://localhost/icinga.

This concludes setup and initial configuration of Icinga.

Icinga and NRPE Quickstart

The following sections provides an overview on monitoring your Uyuni server using Icinga. You will add Uyuni as a host to Icinga and use a Nagios script/plugin to monitor running services via `NRPE` (Nagios Remote Plugin Executor). This section does not attempt to cover all monitoring solutions Icinga has to offer but should help you get started.

Procedure: Adding Uyunito Icinga for Monitoring

1. On your Uyuni server install the required packages:

   ```
   zypper install nagios-nrpe susemanager-nagios-plugin insserv nrpe monitoring-plugins-
   nrpe
   ```

2. Modify the NRPE configuration file located at:

   ```
   /etc/nrpe.cfg
   ```

   Edit or add the following lines:

```
server_port=5666
nrpe_user=nagios
nrpe_group=nagios
allowed_hosts=Icinga.example.com
dont_blame_nrpe=1
command[check_systemd.sh]=/usr/lib/nagios/plugins/check_systemd.sh $ARG1$
```

Variable definitions:

server_port

The variable `server_port` defines the port nrpe will listen on. The default port is 5666. This port must be opened in your firewall.

nrpe_user

The variables `nrpe_user` and `nrpe_group` control the user and group IDs that nrpe will run under. Uyuni probes need access to the database, therefore nrpe requires access to database credentials stored in `/etc/rhn/rhn.conf`. There are multiple ways to achieve this. You may add the user `nagios` to the group `www` (this is already done for other IDs such as tomcat); alternatively you can simply have nrpe run with the effective group ID `www` in `/etc/rhn/rhn.conf`.

allowed_hosts

The variable `allowed_hosts` defines which hosts nrpe will accept connections from. Enter the FQDN or IP address of your Icinga server here.

dont_blame_nrpe

The use of variable `dont_blame_nrpe` is unavoidable in this example. `nrpe` commands by default will not allow arguments being passed due to security reasons. However, in this example you should pass the name of the host you want information on to nrpe as an argument. This action is only possible when setting the variable to 1.

command[check_systemd.sh]

You need to define the command(s) that nrpe can run on Uyuni. To add a new nrpe command specify a command call by adding `command` followed by square brackets containing the actual nagios/icinga plugin name. Next define the location of the script to be called on your Uyuni server. Finally the variable `$ARG1$` will be replaced by the actual host the Icinga server would like information about. In the example above, the command is named `check_systemd.sh`. You can specify any name you like but keep in mind the command name is the actual script stored in `/usr/lib/nagios/plugins/` on your Uyuni server. This name must also match your probe definition on the Icinga server. *This will be described in greater detail later in the chapter. The check_systemd.sh script/plugin will also be provided in a later section.*

3. One your configuration is complete load the new nrpe configuration as root with:

```
systemctl start nrpe
```

This concludes setup of nrpe.

Add a Host to Icinga

To add a new host to Icinga create a host.cfg file for each host in `/etc/icinga/conf.d/`. For example `susemanager.cfg`:

```
define host {
  host_name           susemanager
  alias               SUSE Manager
  address             192.168.1.1
  check_period        24x7
  check_interval      1
  retry_interval      1
  max_check_attempts  10
  check_command       check-host-alive
}
```

Place the host IP address you want to add to Icinga on the `Address` line.

After adding a new host restart Icinga as root to load the new configuation:

```
systemctl restart icinga
```

Adding Services to Icinga

To add services for monitoring on a specific host define them by adding a service definition to your host.cfg file located in `/etc/icinga/conf.d`. For example you can monitor if a systems SSH service is running with the following service definition.

```
define service {
  host_name           susemanager
  use                 generic-service
  service_description SSH
  check_command       check_ssh
  check_interval      60
}
```

After adding any new services restart Icinga as root to load the new configuration:

```
systemctl restart icinga
```

Creating Icinga Hostgroups

You can create hostgroups to simplify and visualize hosts logically. Create a `hostgroups.cfg` file located in `/etc/icinga/conf.d/` and add the following lines:

```
define hostgroup {
  hostgroup_name  ssh_group
  alias           ssh group
  members         susemanager,mars,jupiter,pluto,examplehost4
}
```

The `members` variable should contain the `host_name` from within each host.cfg file you created to represent your hosts. Every time you add an additional host by creating a host.cfg ensure you add the host_name to the members list of included hosts if you want it to be included within a logical hostgroup.

After adding several hosts to a hostgroup restart Icinga as root to load the new configuration:

```
systemctl restart icinga
```

Creating Icinga Servicegroups

You can create logical groupings of services as well. For example if you would like to create a group of essential Uyuni services which are running define them within a `servicegroups.cfg` file placed in `/etc/icinga/conf.d/`:

```
#Servicegroup 1
define servicegroup {
  servicegroup_name     SUSE Manager Essential Services
  alias                 Essential Services
}

#Servicegroup 2
define servicegroup {
  servicegroup_name     Client Patch Status
  alias                 SUSE Manager 3 Client Patch Status
}
```

Within each host's `host.cfg` file add a service to a servicegroup with the following variable:

```
define service {
  use                 generic-service
  service_description SSH
  check_command       check_ssh
  check_interval      60
  servicegroups       SUSE Manager Essential Services
}
```

All services that include the `servicegroups` variable and the name of the servicegroup will be added to the specified servicegroup. After adding services to a servicegroup restart Icinga as root to load the new configuation:

```
systemctl restart icinga
```

Monitoring Systemd Services

The following section provides information on monitoring uptime of critical Uyuni services.

Procedure: Monitoring Running Systemd Services

1. As root create a new plugin file called `check_systemd.sh` in `/usr/lib/nagios/plugins/` on your Uyuni server:

```
vi /usr/lib/nagios/plugins/ check_systemd.sh
```

2. For this example you will use an opensource community script to monitor Systemd services. You may also wish to write your own.

```
#!/bin/bash
# Copyright (C) 2016 Mohamed El Morabity <melmorabity@fedoraproject.com>
#
# This module is free software: you can redistribute it and/or modify it under
# the terms of the GNU General Public License as published by the Free Software
# Foundation, either version 3 of the License, or (at your option) any later
# version.
#
# This software is distributed in the hope that it will be useful, but WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
# FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along with
# this program. If not, see <http://www.gnu.org/licenses/>.

PLUGINDIR=$(dirname $0)
. $PLUGINDIR/utils.sh


if [ $# -ne 1 ]; then
    echo "Usage: ${0##*/} <service name>" >&2
    exit $STATE_UNKNOWN
fi

service=$1

status=$(systemctl is-enabled $service 2>/dev/null)
r=$?
if [ -z "$status" ]; then
    echo "ERROR: service $service doesn't exist"
    exit $STATE_CRITICAL
fi

if [ $r -ne 0 ]; then
    echo "ERROR: service $service is $status"
    exit $STATE_CRITICAL
fi

systemctl --quiet is-active $service
if [ $? -ne 0 ]; then
    echo "ERROR: service $service is not running"
    exit $STATE_CRITICAL
fi

echo "OK: service $service is running"
exit $STATE_OK
```

A current version of this script can be found at: https://github.com/melmorabity/nagios-plugin-systemd-service/blob/master/check_systemd_service.sh

Non-supported 3rd Party Plugin

The script used in this example is an external script and is not supported by SUSE.

Always check to ensure scripts are not modified or contain malicous code before using them on production machines.

3. Make the script executable:

```
chmod 755 check_systemd.sh
```

4. On your SUSE manager server add the following line to the `nrpe.cfg` located at `/etc/nrpe.cfg` :

```
# SUSE Manager Service Checks
command[check_systemd.sh]=/usr/lib/nagios/plugins/check_systemd.sh $ARG1$
```

This will allow the Icinga server to call the plugin via nrpe on Uyuni.

5. Provide proper permissions by adding the script to the sudoers file:

```
visudo
```

```
nagios  ALL=(ALL)       NOPASSWD:/usr/lib/nagios/plugins/check_systemd.sh
Defaults:nagios !requiretty
```

You can also add permissions to the entire plugin directory instead of allowing permissions for individual scripts:

```
nagios  ALL=(ALL)       NOPASSWD:/usr/lib/nagios/plugins/
```

6. On your Icinga server define the following command within `/etc/icinga/objects/commands.cfg` :

```
define command {
        command_name   check-systemd-service
        command_line   /usr/lib/nagios/plugins/check_nrpe -H $HOSTADDRESS$ -c
check_systemd.sh -a $ARG1$
}
```

7. Now you will add the following critical services to be montitored to your Uyuni host file:
   - auditlog-keeper.service
   - jabberd.service
   - spacewalk-wait-for-jabberd.service
   - tomcat.service
   - spacewalk-wait-for-tomcat.service
   - salt-master.service
   - salt-api.service
   - spacewalk-wait-for-salt.service
   - apache2.service
   - osa-dispatcher.service
   - rhn-search.service
   - cobblerd.service
   - taskomatic.service
   - spacewalk-wait-for-taskomatic.service

   On your Icinga server add the following service blocks to your Uyuni host file `susemanager.cfg` file located in `/etc/icinga/conf.d/`. (This configuration file was created in the previous section *Adding a Host to Icinga*.)

```
# Monitor Audit Log Keeper
define service {
        use                     generic-service
        host_name               susemanager
        check_interval          1
        active_checks_enabled   1
        service_description     Audit Log Keeper Service
        servicegroups           SUSE Manager Essential Services
        check_command           check-systemd-service!auditlog-keeper.service

}

# Monitor Jabberd
define service {
        use                     generic-service
        host_name               susemanager
        check_interval          1
        active_checks_enabled   1
        service_description     Jabberd Service
        servicegroups           SUSE Manager Essential Services
        check_command           check-systemd-service!jabberd.service

}

# Monitor Spacewalk Wait for Jabberd
define service{
        use                     generic-service
        host_name               susemanager
        check_interval          1
        active_checks_enabled  1
        service_description    Spacewalk Wait For Jabberd Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!spacewalk-wait-for-
jabberd.service
}

# Monitor Tomcat
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Tomcat Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!tomcat.service
}

# Monitor Spacewalk Wait for Tomcat
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Spacewalk Wait For Tomcat Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!spacewalk-wait-for-
tomcat.service
}

# Monitor Salt Master
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Salt Master Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!salt-master.service
}

# Monitor Salt API
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Salt API Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!salt-api.service
}

# Monitor Spacewalk Wait for Salt
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Spacewalk Wait For Salt Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!spacewalk-wait-for-salt.service
}

# Monitor apache2
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Apache2 Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!apache2.service
}

# Monitor osa dispatcher
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Osa Dispatcher Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!osa-dispatcher.service
}

# Monitor rhn search
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    RHN Search Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!rhn-search.service
}

# Monitor Cobblerd
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Cobblerd Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!cobblerd.service
}

# Monitor taskomatic
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Taskomatic Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!taskomatic.service
}

# Monitor wait for taskomatic
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Spacewalk Wait For Taskomatic Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!spacewalk-wait-for-
taskomatic.service
}
```

Each of these service blocks will be passed as the check-systemd-service!$ARG1$ variable to SUSE manager server via nrpe. You probably noticed the servicegroups parameter was also included. This adds each service to a servicegroup and has been defined in a servicesgroups.cfg file located in /etc/icinga/conf.d/:

```
define servicegroup {
       servicegroup_name     SUSE Manager Essential Services
       alias                 Essential Services
}
```

8. Restart Icinga:

```
systemctl restart icinga
```

Using the check_suma_patches Plugin

You can use the check_suma_patches plugin to check if any machines connected to Uyuni as clients require a patch or an update. The following procedure will guide you through the setup of the check_suma_patches plugin.

Procedure: Setup check_suma_patches

1. On your Uyuni server open /etc/nrpe.cfg and add the following lines:

```
# SUSE Manager check_patches
command[check_suma_patches]=sudo /usr/lib/nagios/plugins/check_suma_patches $ARG1$
```

2. On your Icinga server open /etc/icinga/objects/commands.cfg and define the following command:

```
define command{
        command_name    check_suma
        command_line    /usr/lib/nagios/plugins/check_nrpe -H 192.168.1.1 -c $ARG1$ -a
$HOSTNAME$
}
```

3. On your Icinga server open any of your Uyuni client host configration files located at /etc/icinga/conf.d/clients.cfg and add the following service definition:

```
define service {
        use                             generic-service
        host_name                       client-hostname
        service_description             Available Patches for client-host_name
        servicegroups                   Client Patch Status
        check_command                   check_suma!check_suma_patches
}
```

4. In the above service definition notice that this host is included in the servicegroup labeled *Client Patch Status*. Add the following servicegroup definition to /etc/icinga/conf.d/servicegroups.cfg to create a servicegroup:

```
define servicegroup {
       servicegroup_name     Client Patch Status
       alias                 SUSE Manager 3 Client Patch Status
}
```

5.
   - `OK:System is up to date`
   - `Warning: At least one patch or package update is available`
   - `Critical:At least one security/critical update is available`
   - `Unspecified:The host cannot be found in the SUSE Manager database or the host name is not unique`

This concludes setup of the `check_suma_patches` plugin.

Using the check_suma_lastevent Plugin

You can use the `check_suma_lastevent` plugin to display the last action executed on any host.

The following procedure will guide you through the setup of the check_suma_patches plugin.

Procedure: Setup check_suma_lastevent

1. On your Uyuni server open `/etc/nrpe.cfg` and add the following lines:

```
# Check SUSE Manager Hosts last events
command[check_events]=sudo /usr/lib/nagios/plugins/check_suma_lastevent $ARG1$
```

2. On the Icinga server open `/etc/icinga/objects/commands.cfg` and add the following lines:

```
define command {
        command_name    check_events
        command_line    /usr/lib/nagios/plugins/check_nrpe -H manager.suse.de -c $ARG1$
-a $HOSTNAME$
}
```

3. On your Icinga server add the following line to a host.cfg service definition:

```
define service{
        use                             generic-service
        host_name                       hostname
        service_description             Last Events
        check_command                   check_events!check_suma_lastevent
}
```

4. Status will be reported as follows:
   - `OK:Last action completed successfully`

  - `Warning: Action is currently in progress`
  - `Critical:Last action failed`
  - `Unspecified:The host cannot be found in the Uyuni database or the host name is not unique`

This concludes setup of the `check_suma_lastevent` plugin.

Additional Resources

For more information, see Icinga's official documentation located at http://docs.icinga.org/latest/en.

For some excellent time saving configuration tips and tricks not covered in this guide, see the following section located within the official documentation: http://docs.icinga.org/latest/en/objecttricks.html

Kubernetes

Prerequisites

The prerequisites listed below should be met before proceeding.

- At least one *Kubernetes* or _SUSE CaaS Platform _ cluster available on your network
- Uyuni configured for container management

  Required channels are present, a registered build host available etc.

- virtual-host-gatherer-Kubernetes package installed on your Uyuni server

Requirements

- Kubernetes version 1.5.0 or higher. Alternatively use SUSE CaaS Platform *(SUSE CaaS Platform includes Kubernetes 1.5.0 by default)*
- Docker version 1.12 or higher on the container build host

  To enable all Kubernetes related features within the Web UI, the virtual-host-gatherer-Kubernetes package must be installed.

Register Kubernetes as a Virtual Host Manager

Kubernetes clusters are registered with SUSE Manager as `virtual host managers`. Registration and authorization begins with importing a `kubeconfig` file using Kubernetes official command line tool `kubectl`.

Procedure: Registering a Kubernetes Cluster with Uyuni

1. Select **Systems** › **Virtual Host Managers** from the navigation menu.
2. Expand the `Create` dropdown in the upper right corner of the page and select **Kubernetes Cluster** .
3. Input a label for the new Virtual Host Manager.
4. Select the `kubeconfig` file which contains the required data for the Kubernetes cluster.
5. Select the correct *context* for the cluster, as specified in the kubeconfig file.
6. Click `Create`.

View the List of Nodes in a Cluster

1. Select **Systems** › **Virtual Host Managers** from the navigation menu.
2. Select the desired Kubernetes cluster to view it.

3. Node data is not refreshed during registration. To refresh node data, click on `Schedule refresh data`.
4. Refresh the browser. If the node data is not available wait a few moments and try again.

Obtain Runtime Data about Images

See the following steps to find runtime data for images.

1. Select **Images** › **Images** from the navigation menu.
2. In the image list table, take notice of the new runtime columns. These are labeled: `Revision`, `Runtime` and `Instances`. Initially these columns will not provide useful data.
   - `Revision`: An artificial sequence number which increments on every rebuild for manager-built images, or on every reimport for externally built images.
   - `Runtime`: Overall status of the running instances of the image throughout the registered clusters. The status can be one of the following:
     - All instances are consistent with SUSE Manager: All the running instances are running the same build of the image as tracked by SUSE Manager.
     - Outdated instances found: Some of the instances are running an older build of the image. A redeploy of the image into the pod may be required.
     - No information: The checksum of the instance image does not match the image data contained in SUSE Manager. A redeploy of the image into the pod may be required.
   - `Instances`: Number of instances running this image across all the clusters registered in SUSE Manager. A breakdown of numbers can be seen by clicking on the pop-up icon next to the number.

Build an image for deployment in Kubernetes

The following steps will help you build an image for deployment in Kubernetes.

1. Under **Images** › **Stores**, create an image store.
2. Under **Images** › **Profiles**, create an image profile (with a Dockerfile which is suitable to deploy to Kubernetes).
3. Under **Images** › **Build**, build an image with the new profile and wait for the build to finish.
4. Deploy the image into one of the registered Kubernetes clusters (via `kubectl`).
5. Notice the updated data in `Runtime` and `Instances` columns in the respective image row.

Import a Previously Deployed Image in Kubernetes

The following steps will guide you through importing a previously deployed image in Kubernetes.

1. Select an image that has already been deployed to any of your registered Kubernetes clusters.
2. Add the registry owning the image to SUSE Manager as an image store.
3. Select **Images** › **Images** , click `Import` from the top-right corner, fill in the form fields and click `Import`.
4. Notice the updated data in `Runtime` and `Instances` columns in the respective image row.

Obtain Additional Runtime Data

The following steps will help you find additional runtime data.

1. Select to **Images** › **Images** , click the `Details` button on the right end of a row which has running instances.
2. Under the `Overview` tab, notice the data in `Runtime` and `Instances` fields under `Image Info` section.
3. Select the `Runtime` tab.
4. Here is a breakdown of the Kubernetes pods running this image in all the registered clusters including the following data:
   - Pod name
   - Namespace which the pod resides in
   - The runtime status of the container in the specific pod. Status icons are explained in the preceeding example.

Rebuild a Previously Deployed Image in Kubernetes

The following steps will guide you through rebuilding an image which has been deployed to a Kubernetes cluster.

1. Go to **Images** › **Images** , click the Details button on the right end of a row which has running instances. The image must be manager-built.
2. Click the `Rebuild` button located under the `Build Status` section and wait for the build to finish.
3. Notice the change in the `Runtime` icon and title, reflecting the fact that now the instances are running a previous build of the image.

Role Based Access Control Permissions and Certificate Data

Currently, only kubeconfig files containing all embedded certificate data may be used with SUSE Manager

The API calls from Uyuni are:

- GET /api/v1/pods
- GET /api/v1/nodes

According to this list, the minimum recommended permissions for Uyuni should be as follows:

- A ClusterRole to list all the nodes:

  ```
  resources: ["nodes"]
  verbs: ["list"]
  ```

- A ClusterRole to list pods in all namespaces (role binding must not restrict the namespace):

  ```
  resources: ["pods"]
  verbs: ["list"]
  ```

Due to a a 403 response from /pods, the entire cluster will be ignored by SUSE Manager.

For more information on working with RBAC Authorization see: https://kubernetes.io/docs/admin/authorization/rbac/

Virtualization

Uyuni allows you to autoinstall and manage Xen and KVM VM Guests on a registered VM Host Server. To autoinstall a VM Guest, an autoinstallable distribution and an autoinstallation profile (AutoYaST or Kickstart) need to exist on Uyuni. VM Guests registered with Uyuni can be managed like “regular” machines. In addition, basic VM Guest management tasks such as (re)starting and stopping or changing processor and memory allocation can be carried out using Uyuni.

The following documentation is valid in the context of traditional clients. Salt minions must be treated differently:

- Autoinstallation is not supported, but creation of a guest from a template disk image is supported.

Limitation to Xen and KVM Guests

Autoinstalling and managing VM Guests via Uyuni is limited to Xen and KVM guests. Uyuni uses `libvirt` for virtual machine management. Currently, virtual machines from other virtualization solutions such as VMware* or VirtualBox*, are recognized as VM Guests, but cannot be managed from within Uyuni.

Autoinstalling VM Guests

With Uyuni you can automatically deploy Xen and KVM VM Guests using AutoYaST or Kickstart profiles. It is also possible to automatically register the VM Guests, so they can immediately be managed by Uyuni.

Requirements on Uyuni

Setting up and managing VM Guests with Uyuni does not require special configuration options. However, you need to provide activation keys for the VM Host Server and the VM Guests, an autoinstallable distribution and an autoinstallation profile. To automatically register VM Guests with Uyuni, a bootstrap script is needed.

Activation Keys

Just like any other client, VM Host Server and VM Guests need to be registered with Uyuni using activation keys. Find details on how to set up activation keys at [create.act.keys]. While there are no special requirements for a VM Guest key, at least the following requirements must be met for the VM Host Server activation key.

VM Host Server Activation Key: Minimum Requirements

- Entitlements: Provisioning, Virtualization Platform.
- Packages: `rhn-virtualization-host`, `osad`.

  If you want to manage the VM Host Server system from Uyuni (for example, by executing remote scripts), the package `rhncfg-actions` needs to be installed as well.

Setting up an Autoinstallable Distribution

To autoinstall clients from Uyuni, you need to provide an “autoinstallable distribution”, also referred to as autoinstallable tree or installation source. This installation source needs to be made available through the file system of the Uyuni host. It can for example be a mounted local or remote directory or a “loop-mounted” ISO image. It must match the following requirements:

- Kernel and initrd location:

Red Hat Enterprise Linux / Generic RPM

- images/pxeboot/vmlinuz
- images/pxeboot/initrd.img

SUSE

- boot/`arch`/loader/initrd
- boot/`arch`/loader/linux
  - The **Base Channel** needs to match the autoinstallable distribution.

Autoinstallation package sources

There is a fundamental difference between Red Hat Enterprise Linux and SUSE systems regarding the package sources for autoinstallation. The packages for a Red Hat Enterprise Linux installation are being fetched from the **Base Channel**. Packages for installing SUSE systems are being fetched from the autoinstallable distribution.

As a consequence, the autoinstallable distribution for a SUSE system has to be a complete installation source (same as for a regular installation).

Procedure: Creating Autoinstallable Distribution

1. Make sure an installation source is available from a local directory. The data source can be any kind of network resource, a local directory or an ISO image (which has to be “loop-mounted”). Files and directories must be world readable.
2. Log in to the Uyuni Web UI and navigate to **Systems** › **Autoinstallation** › **Distributions** › **Create Distribution**.
3. Fill out the form `Create Autoinstallable Distribution` as follows:

   `Distribution Label`

   Choose a unique name for the distribution. Only letters, numbers, hyphens, periods, and underscores are allowed; the minimum length is 4 characters. This field is mandatory.

   `Tree Path`

   Absolute local disk path to installation source. This field is mandatory.

`Base Channel`
: Channel matching the installation source. This channel is the package source for non-SUSE installations. This field is mandatory.

`Installer Generation`
: Operating system version matching the installation source. This field is mandatory.

`Kernel Options`
: Options passed to the kernel when booting for the installation. There is no need to specify the `install=` parameter since it will automatically be added. Moreover, the parameters `self_update=0 pt.options=self_update` are added automatically to prevent AutoYaST from updating itself during the system installation. This field is optional.

`Post Kernel Options`
: Options passed to the kernel when booting the installed system for the first time. This field is optional.

4. Save your settings by clicking **[Create Autoinstallable Distribution]**.

To edit an existing `Autoinstallable Distribution`, go to **Systems** › **Autoinstallation** › **Distributions** and click on a `Label`. Save your settings by clicking **[Update Autoinstallable Distribution]**.

Providing an Autoinstallation Profile

Autoinstallation profiles (AutoYaST or Kickstart files) contain all the installation and configuration data needed to install a system without user intervention. They may also contain scripts that will be executed after the installation has completed.

All profiles can be uploaded to Uyuni and be edited afterwards. Kickstart profiles can also be created from scratch with Uyuni.

A minimalist AutoYaST profile including a script for registering the client with Uyuni is listed in Minimalist AutoYaST Profile for Automated Installations and Useful Enhancements. For more information, examples and HOWTOs on AutoYaST profiles, refer to *SUSE Linux Enterprise AutoYaST* (https://www.suse.com/documentation/sles-12/book_autoyast/data/book_autoyast.html). For more information on Kickstart profiles, refer to your Red Hat Enterprise Linux documentation.

SUSE Linux Enterprise 15 Systems

You need the installation media to setup the distribution. Starting with version 15, there is only one installation media. You will use the same one for SLES, SLED, and all the other SUSE Linux Enterprise 15 based products.

In the AutoYaST profile specify which product is to be installed. For installing SUSE Linux Enterprise Server use the following snippet in `autoyast.xml`:

```
<products config:type="list">
  <listentry>SLES</listentry>
</products>
```

Then then specify all the required modules as `add-on` in `autoyast.xml`. This is a minimal `SLE-Product-SLES15-Pool` selection that will result in a working installation and can be managed by Uyuni:

- SLE-Manager-Tools15-Pool
- SLE-Manager-Tools15-Updates
- SLE-Module-Basesystem15-Pool
- SLE-Module-Basesystem15-Updates
- SLE-Product-SLES15-Updates

It is also recommended to add the following modules:

- SLE-Module-Server-Applications15-Pool
- SLE-Module-Server-Applications15-Updates

Uploading an Autoinstallation Profile

1. Log in to the Uyuni Web interface and open **Systems › Autoinstallation › Profiles › Upload New Kickstart/AutoYaST File**.
2. Choose a unique name for the profile. Only letters, numbers, hyphens, periods, and underscores are allowed; the minimum length is 6 characters. This field is mandatory.
3. Choose an `Autoinstallable Tree` from the drop-down menu. If no `Autoinstallable Tree` is available, you need to add an Autoinstallable Distribution. Refer to Setting up an Autoinstallable Distribution for instructions.
4. Choose a `Virtualization Type` from the drop-down menu. KVM and Xen (para-virtualized and fully-virtualized) are available. Do not choose `Xen Virtualized Host` here.
5. Scroll down to the `File to Upload` dialog, click **[Browse]** to select it, then click **[Upload File]**.
6. The uploaded file will be displayed in the `File Contents` section, where you can edit it.
7. Click **[Create]** to store the profile.

To edit an existing profile, go to **Systems › Autoinstallation › Profiles** and click on a `Label`. Make the desired changes and save your settings by clicking **[Create]**.

Editing existing Kickstart profiles

If you are changing the `Virtualization Type` of an existing Kickstart profile, it may also modify the bootloader and partition options, potentially overwriting any user customizations. Be sure to review the `Partitioning` tab to verify these settings when changing the `Virtualization Type`.

Creating a Kickstart Profile

Currently it is only possible to create autoinstallation profiles for Red Hat Enterprise Linux systems. If installing a SUSE Linux Enterprise Server system, you need to upload an existing AutoYaST profile as described in Uploading an Autoinstallation Profile.

1. Log in to the Uyuni Web interface and go to **Systems** › **Autoinstallation** › **Profiles** › **Create New Kickstart File**.
2. Choose a unique name for the profile. The minimum length is 6 characters. This field is mandatory.
3. Choose a `Base Channel`. This channel is the package source for non-SUSE installations and must match the `Autoinstallable Tree`. This field is mandatory.
4. Choose an `Autoinstallable Tree` from the drop-down menu. If no `Autoinstallable Tree` is available, you need to add an Autoinstallable Distribution. Refer to Setting up an Autoinstallable Distribution for instructions.
5. Choose a `Virtualization Type` from the drop-down menu. KVM and Xen (para-virtualized and fully-virtualized) are available. Do not choose `Xen Virtualized Host` here.
6. Click the **[Next]** button.
7. Select the location of the distribution files for the installation of your VM Guests. There should already be a `Default Download Location` filled out and selected for you on this screen. Click the **[Next]** button.
8. Choose a root password for the VM Guests. Click the **[Finish]** button to generate the profile.

   This completes Kickstart profile creation. After generating a profile, you are taken to the newly-created Kickstart profile. You may browse through the various tabs of the profile and modify the settings as you see fit, but this is not necessary as the default settings should work well for the majority of cases.

Adding a Registration Script to the Autoinstallation Profile

A VM Guest that is autoinstalled does not get automatically registered. Adding a section to the autoinstallation profile that invokes a bootstrap script for registration will fix this. The following procedure describes adding a corresponding section to an AutoYaST profile. Refer to your Red Hat Enterprise Linux documentation for instructions on adding scripts to a Kickstart file.

1. First, provide a bootstrap script on the Uyuni:

- Create a bootstrap script for VM Guests on the Uyuni as described in [generate.bootstrap.script].
- Log in as root to the konsole of Uyuni and go to `/srv/www/htdocs/pub/bootstrap`. Copy `bootstrap.sh` (the bootstrap script created in the previous step) to for example, `bootstrap_vm_guests.sh` in the same directory.
- Edit the newly created file according to your needs. The minimal requirement is to include the activation key for the VM Guests (see Activation Keys for details). We strongly recommend to also include one or more GPG keys (for example, your organization key and package signing keys).

2. Log in to the Uyuni Web interface and go to **Systems** › **Autoinstallation** › **Profiles**. Click on the profile that is to be used for autoinstalling the VM Guests to open it for editing.

   Scroll down to the **File Contents** section where you can edit the AutoYaST XML file. Add the following snippet at the end of the XML file right before the closing `</profile>` tag and replace the given IP address with the address of the Uyuni server. See Minimalist AutoYaST Profile for Automated Installations and Useful Enhancementsfor an example script.

```
<scripts>
  <init-scripts config:type="list">
    <script>
      <interpreter>shell </interpreter>
      <location>
        http://`192.168.1.1`/pub/bootstrap/bootstrap_vm_guests.sh
      </location>
    </script>
  </init-scripts>
</scripts>
```

Only one `<scripts>` section allowed

If your AutoYaST profile already contains a `<scripts>` section, do not add a second one, but rather place the `<script>` part above within the existing `<scripts>` section!

3. Click **Update** to save the changes.

VM Host Server Setup

A VM Host Server system serving as a target for autoinstalling VM Guests from Uyuni must be capable of running guest operating systems. This requires either KVM or Xen being properly set up. For installation instructions for SUSE Linux Enterprise Server systems refer to the *SLES Virtualization Guide* available from https://www.suse.com/documentation/sles-12/book_virt/data/book_virt.html. For instructions on setting up a Red Hat Enterprise Linux VM Host Server refer to your Red Hat Enterprise Linux documentation.

Since Uyuni uses `libvirt` for VM Guest installation and management, the `libvirtd` needs to run on the VM Host Server. The default `libvirt` configuration is sufficient to install and manage VM Guests from Uyuni. However, in case you want to access the VNC console of a VM Guest as a non-root user, you need to configure `libvirt` appropriately. Configuration instructions for `libvirt` on SUSE Linux Enterprise Server are available in the *SLES Virtualization Guide* available from https://www.suse.com/documentation/sles-12/book_virt/data/book_virt.html available from http://www.suse.com/documentation/sles11/. For instructions for a Red Hat Enterprise Linux VM Host Server refer to your Red Hat Enterprise Linux documentation.

Apart from being able to serve as a host for KVM or Xen guests, which are managed by `libvirt`, a VM Host Server must be registered with Uyuni.

1. Make sure either KVM or Xen is properly set up.
2. Make sure the `libvirtd` is running.
3. Register the VM Host Server with Uyuni:
   - Create a bootstrap script on the Uyuni as described in [generate.bootstrap.script].
   - Download the bootstrap script from `susemanager.example.com/pub/bootstrap/bootstrap.sh` to the VM Host Server.
   - Edit the bootstrap script according to your needs. The minimal requirement is to include the activation key for the VM Host Server (see Activation Keys for details). We strongly recommend to also include one or more GPG keys (for example, your organization key and package signing keys).
   - Execute the bootstrap script to register the VM Host Server.

VM Host Server setup on Salt clients

If the VM Host Server is registered as a Salt minion, a final configuration step is needed in order to gather all the guest VMs defined on the VM Host Server:

1. From the **System Details** › **Properties** page, enable the `Add-on System Type Virtualization Host` and confirm with **[Update Properties]**.
2. Schedule a Hardware Refresh. On the **System Details** › **Hardware** page click **[Schedule Hardware Refresh]**.

VM Host Server setup on Traditional clients

Once the registration process is finished and all packages have been installed, enable the `osad` (Open Source Architecture Daemon). On a SUSE Linux Enterprise Server system this can be achieved by running the following commands as user root:

```
systemctl stop rhnsd
systemctl disable rhnsd
```

```
systemctl enable osad
systemctl start osad
```

`osad`*Together with* `rhnsd`

The `rhnsd` daemon checks for scheduled actions every four hours, so it can take up to four hours before a scheduled action is carried out. If many clients are registered with Uyuni, this long interval ensures a certain level of load balancing since not all clients act on a scheduled action at the same time.

However, when managing VM Guests, you usually want actions like rebooting a VM Guest to be carried out immediately. Adding `osad` ensures that. The `osad` daemon receives commands over the jabber protocol from Uyuni and commands are instantly executed. Alternatively you may schedule actions to be carried out at a fixed time in the future (whereas with `rhnsd` you can only schedule for a time in the future plus up to four hours).

Autoinstalling VM Guests

Once all requirements on the Uyuni and the VM Host Server are met, you can start to autoinstall VM Guests on the host. Note that VM Guests will not be automatically registered with Uyuni, therefore we strongly recommend to modify the autoinstallation profile as described in Adding a Registration Script to the Autoinstallation Profile. VM Guests need to be registered to manage them with Uyuni. Proceed as follows to autoinstall a VM Guest.

No parallel Autoinstallations on VM Host Server

It is not possible to install more than one VM Guest at a time on a single VM Host Server. When scheduling more than one autoinstallation with Uyuni make sure to choose a timing, that starts the next installation after the previous one has finished. If a guest installation starts while another one is still running, the running installation will be cancelled.

1. In the Web UI click the **Main Menu** › **Systems** › **Systems** tab.
2. Click the VM Host Server's name to open its `System Status` page.
3. Open the form for creating a new VM Guest by clicking **Virtualization** › **Provisioning**. Fill out the form by choosing an autoinstallation profile and by specifying a name for the VM Guest (must not already exist on VM Host Server). Choose a proxy if applicable and enter a schedule. To change the VM Guest's hardware profile and configuration options, click **[Advanced Options]**.
4. Finish the configuration by clicking **[Schedule Autoinstallation and Finish]**. The `Session Status` page opens for you to monitor the autoinstallation process.

Checking the Installation Log

To view the installation log, click **Events** › **History** on the `Session Status` page. On the `System History Event` page you can click a `Summary` entry to view a detailed log.

In case an installation has failed, you can **[Reschedule]** it from this page once you have corrected the problem. You do not have to configure the installation again.

If the event log does not contain enough information to locate a problem, log in to the VM Host Server console and read the log file for your package manager. If you are using the `rhnsd`, you may alternatively immediately trigger any scheduled actions by calling `rhn_check` on the VM Host Server. Increase the command's verbosity by using the options `-v`, `-vv`, or `-vvv`, respectively.

Managing VM Guests

Basic VM Guest management actions such as restarting or shutting down a virtual machine as well as changing the CPU and memory allocation can be carried out in the Uyuni Web interface if the following requirements are met:

- VM Host Server must be a KVM or Xen host.
- `libvirtd` must be running on VM Host Server.
- VM Host Server must be registered with Uyuni.

In addition, if you want to see the profile of the VM Guest, install packages, etc., you must also register it with Uyuni.

All actions can be triggered in the Uyuni Web UI from the `Virtualization` page of the VM Host Server. Navigate to this page by clicking the **Main Menu** › **Systems** › **Systems**. On the resulting page, click the VM Host Server's name and then on `Virtualization`. This page lists all VM Guests for this host, known to Uyuni.

Displaying a VM Guest 's Profile

Click the name of a VM Guest on the VM Host Server's `Virtualization` page to open its profile page with detailed information about this guest. For details, refer to [ref.webui.systems.systems].

A profile page for a virtual system does not differ from a regular system's profile page. You can perform the same actions (for example, installing software or changing its configuration).

Starting, Stopping, Suspending and Resuming a VM Guest

To start, stop, restart, suspend, or resume a VM Guest, navigate to the VM Host Server's `Virtualization` page. Click the corresponding action button in the row of the VM Guest.

Alternatively, check one or more `Guests` listed in the table and click the corresponding button above the table. **[Confirm]** the action on the displayed popup dialog.

Automatically restarting a VM Guest

Automatically restarting a VM Guest when the VM Host Server reboots is not enabled by default on VM Guests and cannot be configured from Uyuni. Refer to your KVM or Xen documentation. Alternatively, you may use `libvirt` to enable automatic reboots.

Changing the CPU or RAM allocation of a VM Guest

To change the CPU or RAM allocation of a VM Guest navigate to the VM Host Server's `Virtualization` page. Click the **[Edit]** button on the VM Guest row. Change the values to the desired ones in the next page and click the **[Update]** button to apply.

The memory allocation can be changed on the fly, provided the memory ballooning driver is installed on the VM Guest. If this is not the case, or if you want to change the CPU allocation, you need to shutdown the guest first. Refer to Starting, Stopping, Suspending and Resuming a VM Guest for details.

You can also perform more advanced VM Guest editing tasks on Salt minions, such as adding or removing disks and network interfaces, and changing the display type.

Deleting a VM Guest

Deleting a VM Guest is only possible on Salt minions, not on traditional clients.

To delete a VM Guest, navigate to the VM Host Server's `Virtualization` page. Click the **[Delete]** button on the VM Guest row. Alternatively, check one or more `Guests` listed in the table and click the **[Delete]** button above the table. **[Confirm]** the action on the displayed popup dialog.

Displaying VM Guest graphical console

In order to be able to display a VM Guest VNC or Spice graphical console, the virtual host corresponding port needs to be reachable by the server. The VM Guest graphics settings also have to listen on at least the virtual host address. This is the default for any VM Guest created using the web interface.

Virtual Hosts

Inventorying vCenter/vSphere ESXi Hosts with Uyuni

Introduction

Foreign virtual hosts (such as vCenter and vSphere ESXi) can be inventoried using the `Virtual Host Manager`. From the vSphere Client you can define roles and permissions for vCenter and vSphere ESXi users allowing vSphere objects and resources to be imported and inventoried by Uyuni. Objects and resources are then displayed as foreign hosts on the Uyuni **Systems › Virtual Systems** page.

The following sections will guide you through:

- Requirements
- Overview of permissions and roles
- Adding vCenter and vSphere ESXi hosts to Uyuni

Requirements

This table displays the default API communication port and required access rights for inventorying objects and resources:

Ports / Permissions	Description
443	Default port that Uyuni uses to access the ESXi API for obtaining infrastructure data
read-only	All vCenter/ESXi objects and resources that should be inventoried by the Virtual Host Manager should be at least assigned the *read-only* role. Mark objects and resources with *no-access* to exclude them from the inventory.

Permissions and Roles Overview

This section will guide you through assigning user permissions and roles in vCenter/ESXi.

A user is someone who has been authorized to access an ESXi host. The Virtual Host Manager (located on the SUSE Manager server) will inventory ESXi data defined by assigned roles and permissions on a user account.

For example: The user *John* has been assigned the *read-only* access role to all servers and datacenters in his company with one exception. John's account has been assigned the *no-access* role on the company's *Financial Database server*. You decide to use John's user account and add the ESXi host to SUSE Manager. During the inventory the *Financial Database server* will be excluded.

Keep user access roles in mind when planning to add ESXi hosts to SUSE manager. Note that SUSE Manager will not inventory any objects or resources assigned with the *no-access* role on any user account.

User Roles/Permissions

When planning to add new ESXi hosts to SUSE Manager, consider if the roles and permissions assigned users require need to be inventoried by SUSE Manager.

Adding New Users and Assigning Roles

See the official vSphere documentation on adding new users and assigning roles.

- Authentication and User Management

Inventorying vCenter/vSphere ESXi Hosts

This procedure guides you through inventorying a VSphere ESXi host with Uyuni.

1. From the Uyuni Web UI select **Main Menu** › **Systems** › **Virtual Host Managers** from the left navigation bar.
2. From the upper right corner of the *Virtual Host Managers* page select **[Create]** VMWare-based.
3. From the *Add a VMware-based Virtual Host Manager* page complete these fields with your ESXi host data:

   Label

   Custom name for your Virtual Host Manager

   Hostname

   Fully-qualified domain name (FQDN) or host IP address

   Port

   Default ESXi API port

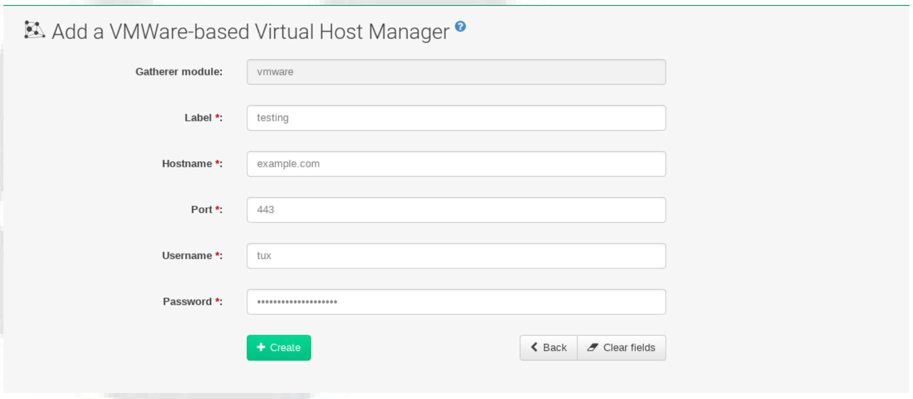
   Username

   Assign a username

   > Remember that only objects and resources which match a user’s defined role will be inventoried. Set the user’s role on objects and resources you want inventoried to *read-only*.

   Password

   ESXi users password



Add a VMWare-based Virtual Host Manager

Gatherer module:	vmware
Label *:	testing
Hostname *:	example.com
Port *:	443
Username *:	tux
Password *:	••••••••••••••••••••

4. Click the **[Create]** button.
5. From the **Systems** › **Virtual Host Managers** page select the new Virtual Host manager.
6. From the **Virtual Host Managers** › **Properties** page click the **[Refresh]** button.

> ! If you do not refresh the data from a new Virtual Host Manager, host data will not be inventoried and therefore will not be displayed under **Systems** › **Virtual Systems**.

7. View inventoried ESXi host objects and resources by selecting **Systems** › **Virtual Systems** .

Security

Setup a Minion to Master Validation Fingerprint

In highly secure network configurations you may wish to ensure your minions are connecting a specific master. To setup validation from minion to master enter the masters fingerprint within the `/etc/salt/minion` configuration file. See the following procedure:

1. On the master enter the following command as root and note the fingerprint:

   ```
   salt-key -F master
   ```

   On your minion, open the minion configuration file located in `/etc/salt/minion`. Uncomment the following line and enter the masters 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 more information on configuring security from a minion see: https://docs.saltstack.com/en/latest/ref/configuration/minion.html

Signing Repository Metadata

Generate a Custom GPG key

To sign repository metadata a custom GPG key is required. Create a new GPG key as **root** via the following steps.

```
$> gpg --gen-key

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: company sign key
Email address: company@example.com
Comment:
You selected this USER-ID:
    "company sign key <company@example.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.

gpg: key 607FABDB marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   2048R/607FABDB 2018-07-09
      Key fingerprint = ACF5 4698 EC70 FD6A F8C9  942B A7A1 9301 607F ABDB
uid       [ultimate] company sign key <company@example.com>
sub   2048R/A812FA62 2018-07-09
```

Configure signing metadata

There are two configuration files which needs to be changed to enable signing of metadata.

1. `/etc/rhn/signing.conf` to specify KEYID and PASSPHRASE
2. `/etc/rhn/rhn.conf` to enable signing metadata

Example for `/etc/rhn/signing.conf`:

```
KEYID="607FABDB"

GPGPASS="MySecretPassword"
```

To enable signing of metadata please add the following in `/etc/rhn/rhn.conf`:

```
sign_metadata = 1
```

All spacewalk services must be restarted after modifying 'rhn.conf'.

Regenerate all metadata

After enabling signing metadata, all metadata needs to be re-generated. This can be done with a small SQL script:

```
$> spacewalk-sql -i
psql (9.6.9)
Type "help" for help.

susemanager=# insert into rhnRepoRegenQueue (id, CHANNEL_LABEL, REASON, FORCE)
susemanager-# (select sequence_nextval('rhn_repo_regen_queue_id_seq'), C.label, 'changed
signing of metadata', 'Y' from rhnChannel C);
INSERT 0 40
susemanager=# \q
$>
```

Trust the GPG key on all clients

When this feature is enabled, all clients have to trust the new GPG key. If the new GPG key is not imported on the client, installation or updating of packages is not possible.

1. Export the GPG key and make it availabel in the 'pub/' directory.

   ```
   # gpg --batch --export -a -o <output filename> <KEYID>
   $> gpg --batch --export -a -o /srv/www/htdocs/pub/company.key 607FABDB
   ```

2. Import the GPG key on all clients

   This can be done using a remote command on all clients

   ```
   # rpm --import http://<server.domain.top/pub/keyname.key
   $> rpm --import http://suma-refhead-srv.mgr.suse.de/pub/company.key
   ```

Tip

For salt managed systems it might make sense to use a state to trust GPG keys.

Authentication Methods

Authentication Via PAM

As security measures become increasingly complex, SUSE Manager supports network-based authentication systems via Pluggable Authentication Modules (PAM). PAM is a suite of libraries that allows to integrate SUSE Manager with a centralized authentication mechanism, thus eliminating the need to remember multiple passwords. SUSE Manager supports LDAP, Kerberos, and other network-based authentication systems via PAM. To enable SUSE Manager to use PAM in your organization's authentication infrastructure, follow the steps below.

1. Set up a PAM service file (default location: `/etc/pam.d/susemanager`) then enforce its use by adding the following line to `/etc/rhn/rhn.conf` :

   ```
   pam_auth_service = susemanager
   ```

   This assumes the PAM service file is named susemanager.

2. To enable a new or existing user to authenticate with PAM, proceed to the **Create User** page and select the checkbox labeled Pluggable Authentication Modules (PAM) positioned below the password and password confirmation fields.

3. To authenticate a SLES system against Kerberos add the following lines to `/etc/pam.d/susemanager` :

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

   To register a Red Hat Enterprise Linux System against Kerberos add the following lines to `/etc/pam.d/susemanager`

```
#%PAM-1.0
 auth        required      pam_env.so
 auth        sufficient    pam_krb5.so no_user_check
 auth        required      pam_deny.so
 account     required      pam_krb5.so no_user_check
```

+

1. YaST can now be used to configure PAM, when packages such as yast2-ldap-client and yast2-kerberos-client are installed; for detailed information on configuring PAM, see the SUSE Linux Enterprise Server Security Guide https://www.suse.com/documentation/sles-12/book_security/data/

part_auth.html. This example is not limited to Kerberos; it is generic and uses the current server configuration. Note that only network based authentication services are supported.

> *Changing Passwords*
>
> Changing the password on the SUSE Manager Web interface changes only the local password on the SUSE Manager server. But this password may not be used at all if PAM is enabled for that user. In the above example, for instance, the Kerberos password will not be changed.

Authentication Via eDirectory and PAM

1. First check to ensure eDirectory authentication is working with your current OS for example:

```
#getent passwd
```

2. If users are returned from eDirectory then create the following file:

```
# cat /etc/pam.d/susemanager
```

3. And add the following content:

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

4. Finally add the following lines to the Uyuni conf file:

```
# grep -i pam /etc/rhn/rhn.conf
 pam_auth_service = susemanager
```

5. You may now create users with the same id that appears on eDirectory and mark the Use PAM check-box from the SUSE Manager WebUI.

Example Quest VAS Active Directory Authentication Template

If you are using Quest VAS for active directory authentication, you can use the following `/etc/pam.d/susemanager` file.

```
#%PAM-1.0
auth       required       pam_env.so
auth       sufficient     pam_vas3.so no_user_check
auth       requisite      pam_vas3.so echo_return
auth       required       pam_deny.so
account    required       pam_vas3.so no_user_check
```

Using a Custom SSL Certificate

The following section will guide you through using a custom certificate with Uyuni 4.0 and SUSE Manager Proxy 4.0.

Prerequisites

The following list provides requirements for using a custom certificate.

- A Certificate Authority (CA) SSL public certificate file
- A Web server SSL private key file
- A Web server SSL public certificate file
- Key and Certificate files must be in PEM format

Hostname and SSL Keys

The hostname of the web server's SSL keys and relevant certificate files must match the hostname of the machine which they will be deployed on.

Intermediate Certificates

In case you want to use CAs with intermediate certificates, merge the intermediate and root CA certificates into one file. It is important that the intermediate certificate comes first within the combined file.

Setup

After completing YaST firstboot procedures, export your current environment variables and point them to the correct SSL files to be imported. Running these commands will make the default certificate obsolete after executing the `yast2 susemanagersetup` command. For more information on YaST firstboot, see https://www.suse.com/documentation/suse-manager-3/singlehtml/suse_manager21/book_susemanager_install/book_susemanager_install.html#sec.manager.inst.setup.

1. Export the environment variables and point to the SSL files to be imported:

```
export CA_CERT=`path_to_CA_certificate_file`export
SERVER_KEY=`path_to_web_server_key`export SERVER_CERT=`path_to_web_server_certificate`
```

2. Execute Uyuni setup with

```
yast2 susemanagersetup
```

Proceed with the default setup. Upon reaching the Certificate Setup window during YaST installation, fill in random values, as these will be overridden with the values specified in [bp.cert.custom.setup.proc.export].

Shell Requirements

Make sure that you execute `yast2 susemanagersetup` from within the same shell the environment variables were exported from.

Using a Custom Certificate with SUSE Manager Proxy

After completing the installation with yast found in [advanced.topics.proxy.quickstart] continue with a modified [at.manager.proxy.run.confproxy] procedure:

1. Execute `configure-proxy.sh`.
2. When prompted with:

   ```
   Do you want to import existing certificates?
   ```

   Answer with `y` .
3. Continue by following the script prompts.

Backup and Restore

Back up your Uyuni installation regularly, in order 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.

> *Backup Space Requirements*
>
> 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.

Backing 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.

If you want to only back up the required files and directories, use this 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.

- `/etc/cobbler/`
- `/etc/dhcp.conf`
- `/etc/fstab` and any ISO mountpoints you require.
- `/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 will also need to have saved a copy of the `id-susemanager` key.
- `/root/ssl-build/`
- `/srv/pillar`
- `/srv/salt`

- `/srv/tftpboot/`
- `/srv/www/cobbler`
- `/srv/www/htdocs/pub/`
- `/var/lib/cobbler/`
- `/var/lib/rhn/kickstarts/`
- `/var/spacewalk/`
- Any directories containing custom data (such as scripts, Kickstart profiles, AutoYaST, and custom RPMs).

> You will also need to back up your database, which you can do by copying `/var/spacewalk/db-backup` or by using the `smdba` tool, which is explained later in this chapter.

Procedure: Restore from a Manual Backup

1. Re-install Uyuni
2. Re-synchronize your installation with the `mgr-sync` tool.
3. 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:

   ```
   rcrhn-search cleanindex
   ```

6. If you did not have `/var/spacewalk/packages/` in your backup, but the source repository still exists, you can restore it by performing a complete channel synchronization.

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 will not work with remotely accessed databases, or Oracle databases.

The `smdba` tool requires `sudo` access, in order 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 the `smdba db-status` command. This command will return either `online` or `offline`:

```
smdba db-status
Checking database core...       online
```

Starting and stopping the database can be performed with `smdba db-start` and `smdba db-stop`

```
smdba db-start
Starting core...       done
```

```
smdba db-stop
Stopping the SUSE Manager database...
Stopping core:         done
```

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 will be created once the files reach this size. Every time you create a new backup, previous backups will be 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.

Performing 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.

When `smdba` is run for the first time, or if you have changed the location of the backup, it will need to restart your database before performing the archive. This will result in a small amount of downtime. Note that regular database backups will not require any downtime.

Procedure: Performing a Manual Database Backup

1. Allocate permanent storage space for your backup. In this procedure, we will be using an NFS share located at `/var/spacewalk/`. This will become a permanent target for your backup, so ensure it will remain 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 /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 run a backup for the first time, run the `smdba backup-hot` command with the `enable` option set. This will create the backup in the specified directory, and, if necessary, restart the database:

   ```
   smdba backup-hot --enable=on --backup-dir=/var/spacewalk/db-backup
   ```
5. Check that the backup files exist in the `/mnt/backup/database` directory, to ensure that your backup has been successful.

Scheduling Automatic Backups

You do not need to shut down your system in order 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.

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:

```
# mkdir /var/spacewalk/db-backup
# chown -R postgres:postgres /var/spacewalk/db-backup
# chmod 700 /var/spacewalk/db-backup
```

2. Open `/etc/cron.d/db-backup-mgr`, or create it if it doesn't 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.

Restoring from Backup

The `smdba` tool can be used to restore from backup in the case of failure.

Procedure: Restoring from Backup

1. Shutdown 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

In SUSE Manager with an embedded database, archive logging is enabled by default. This feature allows the database management tool `smdba` to perform hot backups.

With archive log enabled, even more data is stored on the hard disk:

- Postgresql maintains a limited number of archive logs. Using the default configuration, approx. 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 will generate an additional ~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)

Retrieving 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
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 following 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
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
...
```

Moving the Database

It is possible to move the database to another location. For example if your database storage space is running low. The following procedure will guide you through moving the database to a new location for use by SUSE Manager.

Procedure: Moving the Database

1. The default storage location for SUSE Manager is: `/var/lib/pgsql/` . You would like to move it, for example to: `/storage/postgres/` . To begin, stop the running database with:

   ```
   # rcpostgresql stop
   ```

   Shutdown running spacewalk services with:

   ```
   # spacewalk-service stop
   ```

2. Copy the current working directory structure with the following syntax:

   ```
   cp [OPTION]... SOURCE... DIRECTORY
   ```

   using the `-a, --archive` option. For example:

   ```
   # cp -ar /var/lib/pgsql/ /storage/postgres/
   ```

   This command will copy the contents of `/var/lib/pgsql/` to `/storage/postgres/pgsql/` .

   > The contents of the /var/lib/pgsql needs to remain the same or the SUSE Manager database may malfunction. You also should ensure there is enough available disk space.

3. Mount the new database directory with:

   ```
   # mount /storage/postgres/pgsql
   ```

4. Make sure ownership is `postgres:postgres` and not `root:root` by changing to the new directory and running the following command:

   ```
   /var/lib/pgsql/ # cd /storage/postgres/pgsql/
   /storage/postgres/pgsql/ # l
   total 8
   drwxr-x---  4 postgres postgres   47 Jun  2 14:35 ./
   ```

5. Add the new database mount location to your servers fstab by editing `etc/fstab` .
6. Start the database with:

   ```
   # rcpostgresql start
   ```

   Start spacewalk-services with:

   ```
   # spacewalk-service start
   ```

Recovering from a Crashed Root Partition

This section provides guidance on restoring your server after its root partition has crashed. This section assumes you have setup your server similar to the procedure explained in Getting Started guide with separate partitions for the database and for channels mounted at `/var/lib/pgsql` and `/var/spacewalk/` .

Procedure: Recovering from a Crashed Root Partition

1. Start by installing SLES12 SP2 and the SUSE Manager Extension. Do not mount the `/var/spacewalk` and `/var/lib/pgsql` partitions.
2. Once installation of SUSE Manager has completed shutdown services with `spacewalk-service shutdown` and the database with `rcpostgresql stop`.
3. Mount your `/var/spacewalk` and `/var/lib/pgsql` partitions and restore the directories listed in section one.
4. Start SUSE Manager services and the database with `spacewalk-services start` and `rcpostgresql start`
5. SUSE Manager should now operate normally without loss of your database or synced channels.

Database Connection Information

The information for connecting to the SUSE Manager database is located 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 =
```

Optimization and Scalability

Optimizing Apache and Tomcat

Altering Apache and Tomcat Parameters

Apache and Tomcat Parameters should only be modified with support or consulting as these parameters can have severe and catastrophic performance impacts on your server when improperly adjusted. SUSE will not be able to provide support for catastrophic failure when these advanced parameters are modified without consultation. Tuning values for Apache httpd and Tomcat requires that you align these parameters with your server hardware. Furthermore testing of these altered values should be performed within a test environment.

Apache's httpd MaxClients Parameter

The `MaxClients` setting determines the number of Apache httpd processes, and thus limits the number of client connections that can be made at the same time (SUSE Manager uses the pre-fork MultiProcessing Modules). The default value for `MaxClients` in SUSE Manager is 150. If you need to set the `MaxClients` value greater than 150, Apache httpd's ServerLimit setting and Tomcat's `maxThreads` must also be increased accordingly (see below).

The Apache httpd `MaxClients` parameter must always be less or equal than Tomcat's `maxThreads` parameter!

If the `MaxClients` value is reached while the software is running, new client connections will be queued and forced to wait, this may result in timeouts. You can check the Apache httpd's `error.log` for details:

```
[error] Server reached MaxClients setting, consider increasing the MaxClients setting
```

The default `MaxClients` parameter can be overridden on SUSE Manager by editing the `server-tuning.conf` file located at `/etc/apache2/`. For example `server-tuning.conf` file:

```
# prefork MPM
   <IfModule prefork.c>
           # number of server processes to start
           # http://httpd.apache.org/docs/2.2/mod/mpm_common.html#startservers
           StartServers         5
           # minimum number of server processes which are kept spare
           # http://httpd.apache.org/docs/2.2/mod/prefork.html#minspareservers
           MinSpareServers      5
           # maximum number of server processes which are kept spare
           # http://httpd.apache.org/docs/2.2/mod/prefork.html#maxspareservers
           MaxSpareServers     10
           # highest possible MaxClients setting for the lifetime of the Apache process.
           # http://httpd.apache.org/docs/2.2/mod/mpm_common.html#serverlimit
           ServerLimit        150
           # maximum number of server processes allowed to start
           # http://httpd.apache.org/docs/2.2/mod/mpm_common.html#maxclients
           MaxClients         150
           # maximum number of requests a server process serves
           # http://httpd.apache.org/docs/2.2/mod/mpm_common.html#maxrequestsperchild
           MaxRequestsPerChild  10000
   </IfModule>
```

Whenever the Apache httpd `MaxClients` parameter is changed, the `ServerLimit` must also be updated to the same value, or the change will have no effect.

Tomcat’s maxThreads Parameter

Tomcat’s `maxThreads` represents the maximum number of request processing threads that it will create. This value determines the maximum number of simultaneous requests that it is able to handle. All HTTP requests to the SUSE Manager server (from clients, browsers, XMLRPC API scripts, etc.) are handled by Apache httpd, and some of them are routed to Tomcat for further processing. It is thus important that Tomcat is able to serve the same amount of simultaneous requests that Apache httpd is able to serve in the worst case. The default value for SUSE Manager is 200 and should always be equal or greater than Apache httpd’s `MaxClients`. The `maxThreads` value is located within the `server.xml` file located at `/etc/tomcat/`.

Example relevant lines in `server.xml`:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8"
address="127.0.0.1" maxThreads="200" connectionTimeout="20000"/>
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8"
address="::1" maxThreads="200" connectionTimeout="20000"/>
```

Tuning Notes

When configuring Apache httpd's `MaxClients` and Tomcat's `maxThreads` parameters you should also take into consideration that each HTTP connection will need one or more database connections. If the RDBMS is not able to serve an adequate amount of connections, issues will arise. See the following equation for a rough calculation of the needed amount of database connections:

```
((3 * java_max) +  apache_max + 60)
```

Where:

- 3 is the number of Java processes the server runs with pooled connections (Tomcat, Taskomatic and Search)
- java_max is the maximum number of connections per Java pool (20 by default, changeable in `/etc/rhn/rhn.conf` via the hibernate.c3p0.max_size parameter)
- apache_max is Apache httpd's `MaxClients`
- 60 is the maximum expected number of extra connections for local processes and other uses

Big Scale Deployment (1000 Minions or More)

In the following sections find considerations about a big scale deployment. In this context, a big scale compromises 1000 minions or more.

General Recommendations

SUSE recommends the following in a big scale Uyuni deployment:

- Uyuni servers should have at least 8 recent x86 cores, 32 GiB of RAM, and, most important, fast I/O devices such as at least an SSD (2 SSDs in RAID-0 are strongly recommended).
- Proxies with many minions (hundreds) should have at least 2 recent x86 cores and 16 GiB of RAM.
- Use one SUSE Manager Proxy per 500-1000 clients. Keep into account that download time depends on network capacity. Here is a rough example calculation with physical link speed of 1 GB/s:

```
400 Megabytes  *        3000       /      119 Megabyte/s        / 60
= 169 Minutes
```

This is:

```
Size of updates * Number of minions / Theoretical download speed / 60
```

- Depending on hardware you can accept hundreds of minion keys.
- Plan time for onboarding minions- at least one hour per 1000 minions.
- It is not recommended onboarding more than approx. 1000 minions directly to the Uyuni server- proxies should be used instead. This is because every minion can use up to 3 TCP connections simultaneously, and too many TCP connections can cause performance issues.
- If the following error appears in output of `dmesg`, you probably have an excessive number of minions attached to a single Uyuni server or proxy for the ARP cache to contain all of their addresses:

  ```
  kernel: neighbour table overflow
  ```

  In that case, increase the ARP cache values via `sysctl`, for example, by adding the following lines to `/etc/sysctl.conf`:

  ```
  net.ipv4.neigh.default.gc_thresh1 = 4096
  net.ipv4.neigh.default.gc_thresh2 = 8192
  net.ipv4.neigh.default.gc_thresh3 = 16384
  net.ipv4.neigh.default.gc_interval = 60
  net.ipv4.neigh.default.gc_stale_time = 120
  ```

Start Small and Scale Up

Always start small and scale up gradually. Keep the server monitored in order to identify possible issues early.

Tuning Proposals

SUSE proposes the following tuning settings in a big scale Uyuni deployment:

- Increase the maximum Tomcat heap memory to face a potentially long queue of Salt return results. Set 8 GiB instead of the current default 1 GiB: parameter `Xmx1G` in `/etc/sysconfig/tomcat` (affects onboarding and Action execution).
- Increase the number of Taskomatic workers, allowing to parallelize work on a high number of separate jobs. Set parameter `org.quartz.threadPool.threadCount = 100` in `/etc/rhn/rhn.conf` (affects onboarding and staging).
- Allow Taskomatic to check for runnable jobs more frequently to reduce latency. Set parameter `org.quartz.scheduler.idleWaitTime = 1000` in `/etc/rhn/rhn.conf` (affects onboarding, staging and Action execution).
- Increase Tomcat's Salt return result workers to allow parallelizing work on a high number of Salt return results. Set parameter `java.message_queue_thread_pool_size = 100` in `/etc/rhn/rhn.conf` (affects patching).
- Increase the number of PostgreSQL connections available to Java applications (Tomcat, Taskomatic) according to the previous parameters, otherwise extra workers will starve waiting for a connection.

Set parameter `hibernate.c3p0.max_size = 150` in `/etc/rhn/rhn.conf` (affects all minion operations). Make sure enough PostgreSQL connections are configured before changing this parameter - refer to `smdba system-check autotuning --help` to get automatic tuning of the PostgreSQL configuration file while changing the number of available connections. Additional manual tuning is usually not necessary but might be required depending on scale and exact use cases.

- Increase the number of Taskomatic's `minion-action-executor` worker threads allowing to parallelize the scheduling of Actions to minions. Set parameter `taskomatic.com.redhat.rhn.taskomatic.task.MinionActionExecutor.parallel_threads = 8` in `/etc/rhn/rhn.conf` (affects all minion operations, especially staging).
- Increase Salt's presence ping timeouts if responses might come back later than the defaults. Set parameters `java.salt_presence_ping_timeout = 20` and `java.salt_presence_ping_gather_job_timeout = 20` in `/etc/rhn/rhn.conf` (affects all minion operations).
- Increase the number of Salt master workers so that more requests can run in parallel (otherwise Tomcat and Taskomatic workers will starve waiting for the Salt API, and Salt will not be able to serve files timely). Set parameter `worker_threads: 100` in `/etc/salt/master.d/susemanager.conf` (affects onboarding and patching).
  - Increase this parameter further if file management states fail with the error "Unable to manage file: Message timed out"
  - Note that Salt master workers can consume significant amounts of RAM (typically about 70 MB per worker). It is recommended to keep usage monitored when increasing this value and to do so in relatively small increments (eg. 20) until failures are no longer produced.
- Disable daily comparison of configuration files. Click on **Admin** › **Task Schedules**, then on the **[compare-configs-default]** link, then on the **[Disable Schedule]** button and finally on **[Delete Schedule]**.
- Increase the maximum heap memory for the search daemon to be able to index many minions. Set 4 GiB instead of the current default 512 MB: add `rhn-search.java.maxmemory=4096` in `/etc/rhn/rhn.conf` (affects background indexing only).

Note that increasing the number of Postgres connections will require more RAM, make sure the Uyuni server is monitored and swap is never used.

Also note the above settings should be regarded as guidelines-they have been tested to be safe but care should be exercised when changing them, and consulting support is highly recommended.

Troubleshooting

This chapter provides guidance on registering cloned systems with SUSE Manager. This includes both Salt and Traditional clients. For more information, see https://www.novell.com/support/kb/doc.php?id=7012170.

Registering Cloned Salt Minions

Procedure: Registering a Cloned Salt Minion with SUSE Manager

1. Clone your system (for example using the existing cloning mechanism of your favorite Hypervisor)

   Quick Tips

   Each step in this section is performed on the cloned system, this procedure does not manipulate the original system, which will still be registered to SUSE Manager. The cloned virtual machine should have a different UUID from the original (this UUID is generated by your hypervisor) or SUSE Manager will overwrite the original system data with the new one.

2. Make sure your machines have different hostnames and IP addresses, also check that /etc/hosts contains the changes you made and the correct host entries.

The next step you take will depend on the Operating System of the clone.

The following scenario can occur after on-boarding cloned Salt minions. If after accepting all cloned minion keys from the onboarding page and you see only one minion on the System Overview page, this is likely due to these machines being clones of the original and using a duplicate machine-id. Perform the following steps to resolve this conflict based on OS.

Procedure: SLES 12 Registering Salt Clones

1. SLES 12: If your machines have the same machine ids then delete the file on each minion and recreate it:

```
# rm /etc/machine-id
# rm /var/lib/dbus/machine-id
# dbus-uuidgen --ensure
# systemd-machine-id-setup
```

Procedure: SLES 11 Registering Salt Clones

1. SLES 11: As there is no systemd machine id, generate one from dbus:

```
# rm /var/lib/dbus/machine-id
# dbus-uuidgen --ensure
```

If your machines still have the same minion id then delete the minion_id file on each minion (FQDN will be used when it is regenerated on minion restart):

```
# rm /etc/salt/minion_id
```

Finally delete accepted keys from Onboarding page and system profile from SUSE Manager, and restart the minion with:

```
# systemctl restart salt-minion
```

You should be able to re-register them again, but each minion will use a different '/etc/machine-id' and should now be correctly displayed on the System Overview page.

Registering Cloned Traditional Systems

This section provides guidance on troubleshooting cloned traditional systems registered via bootstrap.

Procedure: Registering a Cloned System with SUSE Manager (Traditional Systems)

1. Clone your system (using your favorite hypervisor.)

   Quick Tips

   Each step in this section is performed on the cloned system, this procedure does not manipulate the original system, which will still be registered to SUSE Manager. The cloned virtual machine should have a different UUID from the original (this UUID is generated by your hypervisor) or SUSE Manager will overwrite the original system data with the new one.

2. Change the Hostname and IP addresses, also make sure /etc/hosts contains the changes you made and the correct host entries.
3. Stop rhnsd daemon with:

   ```
   # /etc/init.d/rhnsd stop
   ```

   or alternativly:

   ```
   # rcrhnsd stop
   ```

4. Stop osad with:

   ```
   # /etc/init.d/osad stop
   ```

   or alternativly:

```
# rcosad stop
```

5. Remove the osad authentifcation configuration file and the systemid with:

```
# rm -f /etc/sysconfig/rhn/{osad-auth.conf,systemid}
```

The next step you take will depend on the Operating System of the clone.

Procedure: SLES 12 Registering A Cloned Traditional System

1.

   If your machines have the same machine ids then delete the file on each client and recreate it:

```
# rm /etc/machine-id
# rm /var/lib/dbus/machine-id
# dbus-uuidgen --ensure
# systemd-machine-id-setup
```

2. Remove the following credential files:

```
# rm  -f /etc/zypp/credentials.d/{SCCcredentials,NCCcredentials}
```

3. Re-run the bootstrap script. You should now see the cloned system in SUSE Manager without overwriting the system it was cloned from.

Procedure: SLES 11 Registering A Cloned Traditional System

1. Continued from section 1 step 5:

```
# suse_register -E
```

   (--erase-local-regdata, Erase all local files created from a previous executed registration. This option make the system look like never registered)

2. Re-run the bootstrap script. You should now see the cloned system in SUSE Manager without overwriting the system it was cloned from.

Procedure: SLES 10 Registering A Cloned Traditional System

1. Continued from section 1 step 5:

```
# rm -rf /etc/{zmd,zypp}
```

2.

```
# ¡¡¡¡ everthing in /var/lib/zypp/ except /var/lib/zypp/db/products/ !!!!!
# check whether this command works for you
# rm -rf /var/lib/zypp/!(db)
```

3.

```
# rm -rf /var/lib/zmd/
```

4. Re-run the bootstrap script. You should now see the cloned system in SUSE Manager without overwriting the system it was cloned from.

Procedure: RHEL 5,6 and 7

1. Continued from section 1 step 5:

```
# rm  -f /etc/NCCcredentials
```

2. Re-run the bootstrap script. You should now see the cloned system in SUSE Manager without overwriting the system it was cloned from.

Typical OSAD/jabberd Challenges

This section provides answers for typical issues regarding OSAD and jabberd.

Open File Count Exceeded

`SYMPTOMS`: OSAD clients cannot contact the SUSE Manager Server, and jabberd requires long periods of time to respond on port 5222.

`CAUSE`: The number of maximum files that a jabber user can open is lower than the number of connected clients. Each client requires one permanently open TCP connection and each connection requires one file handler. The result is jabberd begins to queue and refuse connections.

`CURE`: Edit the `/etc/security/limits.conf` to something similar to the following: `jabbersoftnofile<#clients + 100> jabberhardnofile<#clients + 1000>`

This will vary according to your setup. For example in the case of 5000 clients: `jabbersoftnofile5100 jabberhardnofile6000`

Ensure you update the `/etc/jabberd/c2s.xml` max_fds parameter as well. For example: `<max_fds>6000</max_fds>`

`EXPLANATION`: The soft file limit is the limit of the maximum number of open files for a single process. In SUSE Manager 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 the jabber user, and accounts for open files from the router, s2s and sm processes additionally.

jabberd Database Corruption

`SYMPTOMS`: After a disk is full error or a disk crash event, the jabberd database may have become corrupted. jabberd may then fail to start during spacewalk-service start:

```
Starting spacewalk services...
   Initializing jabberd processes...
       Starting router                                                     done
       Starting sm startproc:  exit status of parent of /usr/bin/sm: 2
failed
   Terminating jabberd processes...
```

/var/log/messages shows more details:

```
jabberd/sm[31445]: starting up
jabberd/sm[31445]: process id is 31445, written to /var/lib/jabberd/pid/sm.pid
jabberd/sm[31445]: loading 'db' storage module
jabberd/sm[31445]: db: corruption detected! close all jabberd processes and run db_recover
jabberd/router[31437]: shutting down
```

`CURE`: Remove the jabberd database and restart. Jabberd will automatically re-create the database:

```
spacewalk-service stop
 rm -Rf /var/lib/jabberd/db/*
 spacewalk-service start
```

An alternative approach would be to test another database, but SUSE Manager does not deliver drivers for this:

```
rcosa-dispatcher stop
 rcjabberd stop
 cd /var/lib/jabberd/db
 rm *
 cp /usr/share/doc/packages/jabberd/db-setup.sqlite .
 sqlite3 sqlite.db < db-setup.sqlite
 chown jabber:jabber *
 rcjabberd start
 rcosa-dispatcher start
```

Capturing XMPP Network Data for Debugging Purposes

If you are experiencing bugs regarding OSAD, it can be useful to dump network messages in order to help with debugging. The following procedures provide information on capturing data from both the client and server side.

Procedure: Server Side Capture

1. Install the tcpdump package on the SUSE Manager Server as root: `zypper in tcpdump`

2. Stop the OSA dispatcher and Jabber processes with `rcosa-dispatcher stop` and `rcjabberd stop.`
3. Start data capture on port 5222: `tcpdump -s 0 port 5222 -w server_dump.pcap`
4. Start the OSA dispatcher and Jabber processes: `rcosa-dispatcher start` and `rcjabberd start`.
5. Open a second terminal and execute the following commands: `rcosa-dispatcher start` and `rcjabberd start`.
6. Operate the SUSE Manager server and clients so the bug you formerly experienced is reproduced.
7. Once you have finished your capture re-open terminal 1 and stop the capture of data with: `CTRL+c`

Procedure: Client Side Capture

1. Install the tcpdump package on your client as root: `zypper in tcpdump`
2. Stop the OSA process: `rcosad stop`.
3. Begin data capture on port 5222: `tcpdump -s 0 port 5222 -w client_client_dump.pcap`
4. Open a second terminal and start the OSA process: `rcosad start`
5. Operate the SUSE Manager server and clients so the bug you formerly experienced is reproduced.
6. Once you have finished your capture re-open terminal 1 and stop the capture of data with: `CTRL+c`

Engineering Notes: Analyzing Captured Data

This section provides information on analyzing the previously captured data from client and server.

1. Obtain the certificate file from your SUSE Manager server: /etc/pki/spacewalk/jabberd/server.pem
2. Edit the certificate file removing all lines before `----BEGIN RSA PRIVATE KEY-----`, save it as key.pem
3. Install Wireshark as root with: `zypper in wireshark`
4. Open the captured file in wireshark.
5. From **Eidt ›]menu:Preferences[** select SSL from the left pane.
6. Select RSA keys list: **Edit ›]menu:New[**
   - IP Address any
   - Port: 5222
   - Protocol: xmpp
   - Key File: open the key.pem file previously edited.
   - Password: leave blank

For more information see also:

- https://wiki.wireshark.org/SSL
- https://bugs.wireshark.org/bugzilla/show_bug.cgi?id=3444

Gathering Information with `spacewalk-report`

The `spacewalk-report` command is used to produce a variety of reports for system administrators. These reports can be helpful for taking inventory of your entitlements, subscribed systems, users, and organizations. Using reports is often simpler than gathering information manually from the SUSE Manager Web UI, especially if you have many systems under management.

`spacewalk-reports` *Package*

To use spacewalk-report, you must have the `spacewalk-reports` package installed.

`spacewalk-report` allows administrators to organize and display reports about content, systems, and user resources across SUSE Manager. Using `spacewalk-report`, you can receive reports on:

1. System Inventory: lists all of the systems registered to SUSE Manager.
2. Entitlements: lists all organizations on SUSE Manager, sorted by system or channel entitlements.
3. Patches: lists all the patches relevant to the registered systems and sorts patches by severity, as well as the systems that apply to a particular patch.
4. Users: lists all the users registered to SUSE Manager and any systems associated with a particular user.

`spacewalk-report` allows administrators to organize and display reports about content, systems, and user resources across SUSE Manager. To get the report in CSV format, run the following at the command line of your SUSE Manager server.

```
spacewalk-report report_name
```

The following reports are available:

Table 1. `spacewalk-report` *Reports*

Report	Invoked as	Description
Channel Packages	`channel-packages`	List of packages in a channel.
Channel Report	`channels`	Detailed report of a given channel.
Cloned Channel Report	`cloned-channels`	Detailed report of cloned channels.

Report	Invoked as	Description
Custom Info	`custom-info`	System custom information.
Entitlements	`entitlements`	Lists all organizations on SUSE Manager with their system or channel entitlements.
Patches in Channels	`errata-channels`	Lists of patches in channels.
Patches Details	`errata-list`	Lists all patches that affect systems registered to SUSE Manager.
All patches	`errata-list-all`	Complete list of all patches.
Patches for Systems	`errata-systems`	Lists applicable patches and any registered systems that are affected.
Host Guests	`host-guests`	List of host-guests mapping.
Inactive Systems	`inactive-systems`	List of inactive systems.
System Inventory	`inventory`	List of systems registered to the server, together with hardware and software information.
Kickstart Trees	`kickstartable-trees`	List of kickstartable trees.
All Upgradable Versions	`packages-updates-all`	List of all newer package versions that can be upgraded.
Newest Upgradable Version	`packages-updates-newest`	List of only newest package versions that can be upgraded.
Result of SCAP	`scap-scan`	Result of OpenSCAP sccdf eval.
Result of SCAP	`scap-scan-results`	Result of OpenSCAP sccdf eval, in a different format.
System Data	`splice-export`	System data needed for splice integration.
System Groups	`system-groups`	List of system groups.
Activation Keys for System Groups	`system-groups-keys`	List of activation keys for system groups.
Systems in System Groups	`system-groups-systems`	List of systems in system groups.
System Groups Users	`system-groups-users`	Report of system groups users.
Installed Packages	`system-packages-installed`	List of packages installed on systems.
Users in the System	`users`	Lists all users registered to SUSE Manager.
Systems administered	`users-systems`	List of systems 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. The description and list of possible fields in the report will be shown.

For further information on program invocations and options, see the `spacewalk-report(8)` man page as well as the `--help`parameter of the `spacewalk-report`.

RPC Connection Timeout Settings

RPC connection timeouts are configurable on the Uyuni server, SUSE Manager Proxy server, and the clients. For example, if package downloads take longer then expected, you can increase timeout values. `spacewalk-proxy restart` should be run after the setting is added or modified.

Set the following variables to a value in seconds specifying how long an RPC connection may take at maximum:

Server -`/etc/rhn/rhn.conf`

```
server.timeout =`number`
```

Proxy Server -`/etc/rhn/rhn.conf`

```
proxy.timeout =`number`
```

SUSE Linux Enterprise Server Clients (using zypp-plugin-spacewalk) -`/etc/zypp/zypp.conf`

```
## Valid values:  [0,3600]
## Default value: 180
download.transfer_timeout = 180
```

This is the maximum time in seconds that a transfer operation is allowed to take. This is useful for preventing batch jobs from hanging for hours due to slow networks or links going down. If limiting operations to less than a few minutes, you risk aborting perfectly normal operations.

Red Hat Enterprise Linux Clients (using yum-rhn-plugin) -`/etc/yum.conf`

```
timeout =`number`
```

Client/Server Package Inconsistency

In some cases, updates are available in the web interface, but not appearing on the client. If you schedule an update on the client, it will fail with an error stating that no updates are available. This can be caused by a metadata regeneration problem, or because update packages have been locked.

The notice that updates are available will appear immediately, but new metadata is only generated on the server after synchronizing. In this case, an inconsistency can occur if taskomatic crashes, or because taskomatic is still running and creating new metadata.

To address this issue, check the `/var/log/rhn/rhn_taskomatic_daemon.log` file to determine if any processess are still running, or an exception which could indicate a crash. In the case of a crash, restart taskomatic.

Check package locks and exclude lists to determine if packages are locked or excluded on the client:

On Expanded Support Platform, check `/etc/yum.conf` and search for `exclude=`.

On SUSE Linux Enterprise Server, use the `zypper locks` command.

Corrupted Repository Data

If the information in `/var/cache/rhn/repodata/sles12-sp3-updates-x86_64` becomes out of date, it will cause problems with updating the server. The repository data file can be regenerated using the `spacemd` command:

Procedure: Rebuild repodata file

1. Remove all files from `/var/cache/rhn/repodata/sles12-sp3-updates-x86_64`
2. Regenerate the file with `spacecmd softwarechannel_regenerateyumcache sles12-sp3-updates-x86_64`

Unable to Get Local Issuer Certificate

Some older bootstrap scripts will will create a link to the local certificate in the wrong place, which can cause problems with zypper returning an `Unrecognized error` about the local issuer certificate. In this case, ensure that the link to the local issuer certificate has been created in `/etc/ssl/certs/`, and consider updating your bootstrap scripts.

AutoYast Example File

Minimalist AutoYaST Profile for Automated Installations and Useful Enhancements

The AutoYaST profile in this section installs a SUSE Linux Enterprise Server system with all default installation options including a default network configuration using DHCP. After the installation is finished, a bootstrap script located on the Uyuni server is executed in order to register the freshly installed system with Uyuni. You need to adjust the IP address of the Uyuni server, the name of the bootstrap script, and the root password according to your environment:

```
<user>
 ...
 <username>root</username>
 <user_password>`linux`</user_password>
</user>

<location>http://`192.168.1.1`/pub/bootstrap/`my_bootstrap.sh`</location>
```

The complete AutoYaST file:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns"
         xmlns:config="http://www.suse.com/1.0/configns">
 <general>
  <mode>
   <confirm config:type="boolean">false</confirm>
  </mode>
 </general>
 <networking>
  <keep_install_network config:type="boolean">true</keep_install_network>
 </networking>
 <software>
  <install_recommended config:type="boolean">true</install_recommended>
   <patterns config:type="list">
    <pattern>base</pattern>
   </patterns>
 </software>
 <users config:type="list">
  <user>
   <encrypted config:type="boolean">false</encrypted>
   <fullname>root</fullname>
   <gid>0</gid>
   <home>/root</home>
   <password_settings>
    <expire></expire>
    <flag></flag>
    <inact></inact>
    <max></max>
    <min></min>
    <warn></warn>
   </password_settings>
   <shell>/bin/bash</shell>
   <uid>0</uid>
   <username>root</username>
   <user_password>linux</user_password>
  </user>
 </users>
 <scripts>
  <init-scripts config:type="list">
   <script>
    <interpreter>shell</interpreter>
    <location>http://192.168.1.1/pub/bootstrap/my_bootstrap.sh</location>
   </script>
  </init-scripts>
 </scripts>
</profile>
```

Use this enhancement fragment to add child channels:

```
<add-on>
 <add_on_products config:type="list">
  <listentry>
   <ask_on_error config:type="boolean">true</ask_on_error>
   <media_url>http://$c_server/ks/dist/child/`channel-label`/`distribution-label`</media_url>
   <name>$c_name</name>
   <product>$c_product</product>
   <product_dir>/</product_dir>
  </listentry>
...
 </add_on_products>
</add-on>
```

Replace `channel-label` and `distribution-label` with the correct labels (such as `sles11-sp1-updates-x86_64` and `sles11-sp2-x86_64`). Ensure that the distribution label corresponds to the Autoinstallable Distribution. Set the variables (such as `$c_server`) according to your environment. For information about variables, see [s4-sm-system-kick-dist-variables].

Add the Updates Channel

It is required that you add the updates tools channel to the <add-on> AutoYaST snippet section. This ensures your systems are provided with an up-to-date version of the `libzypp` package. If you do not include the updates tools channel, you will encounter `400` errors. In this example, the (DISTRIBUTION_NAME) is replaced with the name of the autoinstallation distribution, as created previously, from **Systems** › **Autoinstallation** › **Distributions**

```
<listentry>
     <ask_on_error config:type="boolean">true</ask_on_error>
     <media_url>http://$redhat_management_server/ks/dist/child/sles12-
sp2-updates-x86_64/(DISTRIBUTION_NAME)</media_url>
     <name>sles12 sp2 updates</name>
     <product>SLES12</product>
     <product_dir>/</product_dir>
  </listentry>
```