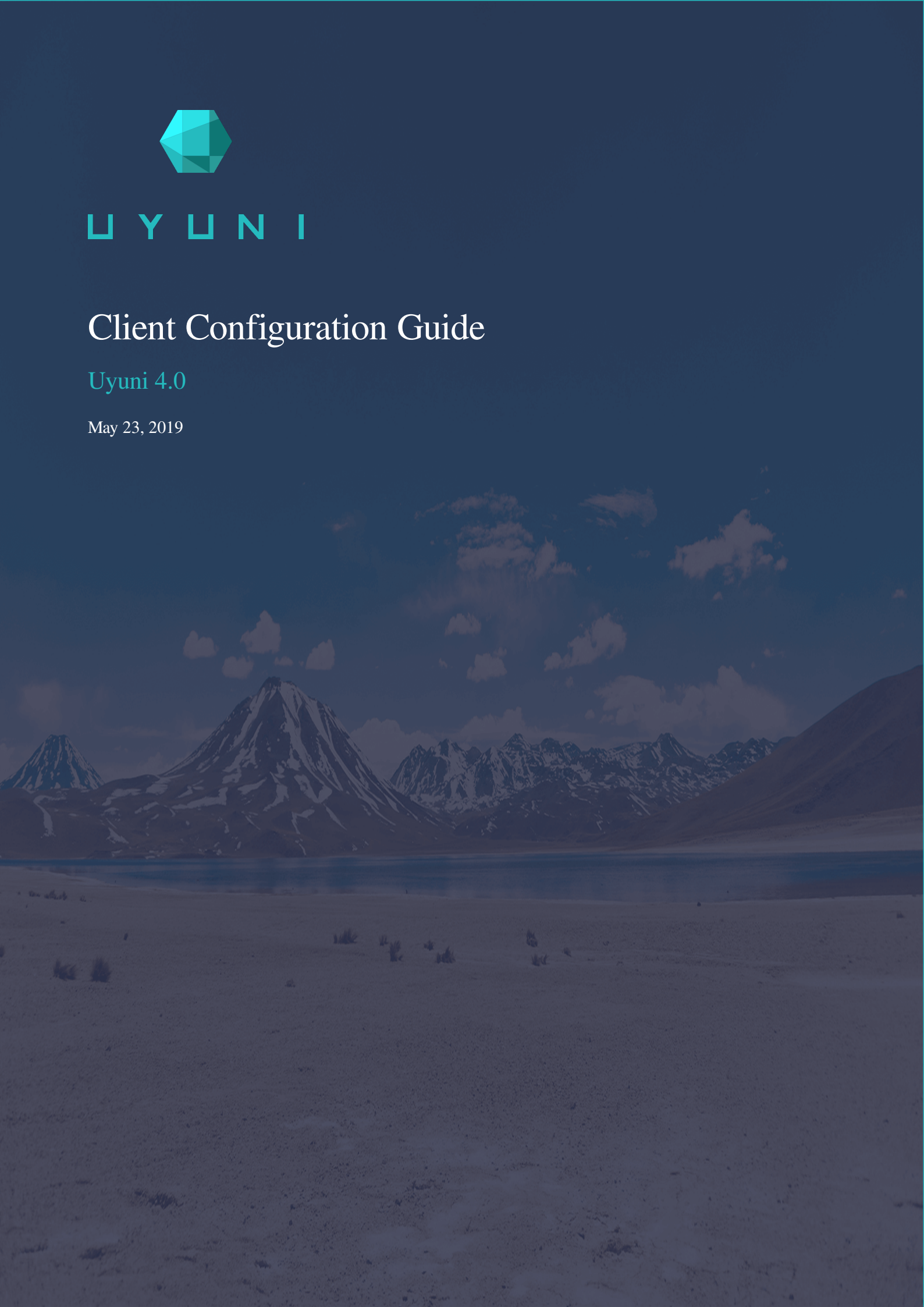


Client Configuration Guide

Uyuni 4.0

May 23, 2019

Table of Contents

Introduction

Registering clients is the first step after installing Uyuni, and most of the time you spend with Uyuni will be spent on maintaining those clients.

Uyuni is compatible with a range of client technologies: you can install traditional or Salt clients, running SUSE Linux Enterprise or another Linux operating system, with a range of hardware options.

This guide discusses how to register and configure different clients, both manually and automatically.

Client Registration Overview

There are two ways to register clients to your Uyuni Server.

For salt clients, the simplest method is to register your clients using the Uyuni Web UI.

If you need to have more control over the process, want to register many clients, or are registering traditional clients, we recommend using a bootstrap script.

Both methods are described in this manual.

Registering Clients with the Web UI

The simplest method to register clients is using the Uyuni Web UI. This method works for Salt clients only.

Procedure: Registering clients in the Web UI

1. In the Uyuni Web UI, navigate to **Systems › Bootstrapping**.
2. In the `Host` field, type the fully-qualified domain name (FQDN) of the client to be bootstrapped.
3. In the `SSH Port` field, type the SSH port number that will be used to connect and bootstrap the client. By default, the SSH port is `22`.
4. In the `User` field, type the username to log in to the client. By default, the username is `root`.
5. In the `Password` field, type password to log in to the client.
6. In the `Activation Key` field, select the activation key that is associated with the software channel you want to use to bootstrap the client.
7. By default, the `Disable SSH Strict Key Host Checking` checkbox is selected. This allows the bootstrap process to automatically accept SSH host keys without requiring you to manually authenticate.
8. OPTIONAL: Check the `Manage System Completely via SSH` checkbox. If you check this option, the client will be configured to use SSH for its connection to the Server, and no other connection method will be configured.
9. Click **[Bootstrap]** to begin registration. When the bootstrap process has completed, your client will be listed at **Systems › System List**.

Registering Clients with a Bootstrap Script

Registering your clients with a bootstrap script gives you more control over parameters, and can help if you have to register a large number of clients at once. This method works for both Salt and traditional clients.

To register clients using a bootstrap script, we recommend you create a template bootstrap script to begin, which can then be copied and modified. The bootstrap script you create is executed on the client when it

is registered, and ensures all the necessary packages are deployed to the client. There are some parameters contained in the bootstrap script which ensure the client system can be assigned to its base channel, including activation keys, and GPG keys.

It is important that you check the repository information carefully, to ensure it matches the base channel repository. If the repository information does not match exactly, the bootstrap script will not be able to download the correct packages.

GPG Keys and Uyuni Client Tools

The GPG key used by Uyuni Client Tools is not trusted by default. When you create your bootstrap script, add a path to the file containing the public key fingerprint with the `ORG_GPG_KEY` parameter.

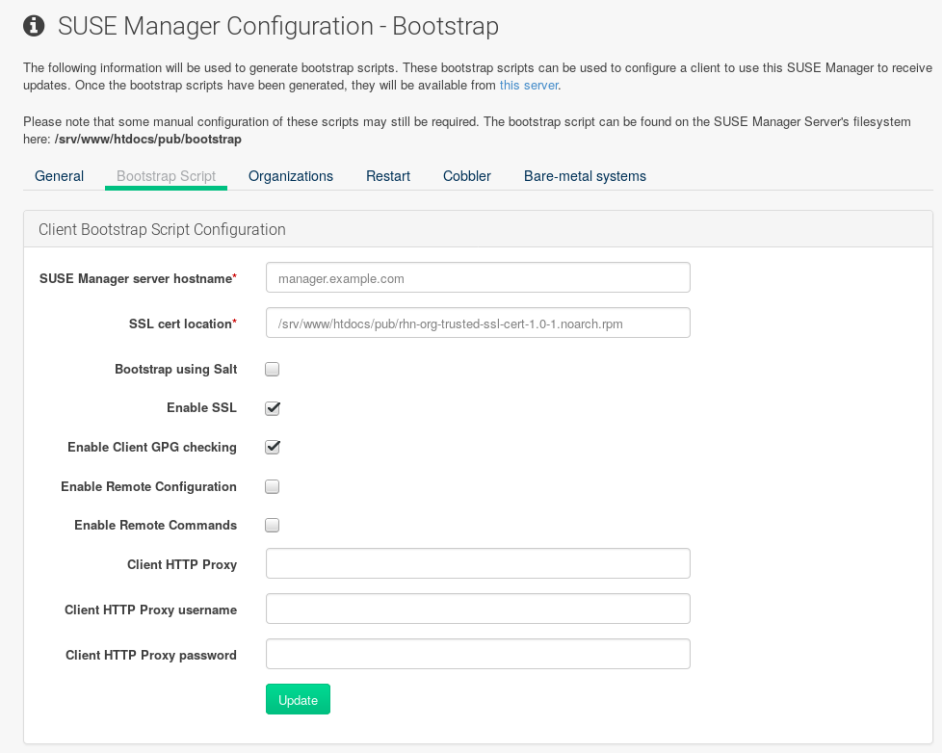
SLES 15 and Python 3

SLES 15 uses Python 3 by default. Bootstrap scripts based on Python 2 must be re-created for SLES 15 systems. Attempting to register SLES 15 systems with using Python 2 bootstrap scripts will fail.

This procedure describes how to generate a bootstrap script.

Procedure: Creating a Bootstrap Script

1. In the Uyuni Web UI, navigate to **Admin › Manager Configuration › Bootstrap Script**.
2. In the `SUSE Manager Configuration - Bootstrap` dialog, uncheck the `Bootstrap using Salt` checkbox if you are installing a traditional client. For Salt clients, leave it checked. Use default settings and click the **[Update]** button.



Using SSL

Unchecking **Enable SSL** in the Web UI or setting `USING_SSL=0` in the bootstrap script is not recommended. If you disable SSL nevertheless you will need to manage custom CA certificates to be able to run the registration process successfully.

3. A template bootstrap script is generated and stored on the server's file system in the `/srv/www/htdocs/pub/bootstrap` directory.

```
cd /srv/www/htdocs/pub/bootstrap
```

The bootstrap script is also available at `https://example.com/pub/bootstrap/bootstrap.sh`.

Editing the Bootstrap Script

In this section you will copy and modify the template bootstrap script you created from xref:FILENAME.adoc#generate.bootstrap.script[].

A minimal requirement when modifying a bootstrap script for use with Uyuni is the inclusion of an activation key.

Most packages are signed with GPG, so you will also need to have trusted GPG keys on your system to install them.

In this procedure, you will need to know the exact name of your activation keys. Navigate to **Home** › **Overview** and click on `Manage Activation keys`. All keys created for channels are listed on this page. You must enter the full name of the key you wish to use in the bootstrap script exactly as presented in the key field.

Procedure: Modifying the Bootstrap Script

1. Login as root from the command line on your Uyuni server.
2. Navigate to the bootstrap directory with:

```
cd /srv/www/htdocs/pub/bootstrap/
```

3. Create and rename two copies of the template bootstrap script for use with each of your clients.

```
cp bootstrap.sh bootstrap-sles11.sh
cp bootstrap.sh bootstrap-sles12.sh
```

4. Open `sles12.sh` for modification. Scroll down and modify both lines marked in green. You must comment out `exit 1` with a hash mark (`#`) to activate the script and then enter the name of the key

for this script in the `ACTIVATION_KEYS=` field as follows:

```
echo "Enable this script: comment (with #'s) this block (or, at least just"
echo "the exit below)"
echo
#exit 1

# can be edited, but probably correct (unless created during initial install):
# NOTE: ACTIVATION_KEYS *must* be used to bootstrap a client machine.
ACTIVATION_KEYS=1-sles12
ORG_GPG_KEY=
```

5. Once you have completed your modifications save the file and repeat this procedure for the second bootstrap script.

Connecting Clients

When you have finished creating your script, you can use it to register clients.

Procedure: Running the Bootstrap Script

1. On the Uyuni Server, log in as root at the command prompt, and navigate to this directory:

```
cd /srv/www/htdocs/pub/bootstrap/
```

2. Run this command to execute the bootstrap script on the client:

```
cat MODIFIED-SCRIPT.SH | ssh root@example.com /bin/bash
```

   The script will execute and proceed to download the required dependencies located in the repositories directory you created earlier.

3. When the script has finished running, you can check that your client is registered correctly by opening the Uyuni Web UI and navigating to **Systems › Overview** to ensure the new client is listed.

Package Locks

Package locks can only be used on traditional clients that use the Zypper package manager. The feature is not currently supported on Red Hat Enterprise Linux or Salt clients.

Package locks are used to prevent unauthorized installation or upgrades to software packages on traditional clients. When a package has been locked, it will show a padlock icon, indicating that it can not be installed. Any attempt to install a locked package will be reported as an error in the event log.

Locked packages can not be installed, upgraded, or removed, either through the Uyuni Web UI, or directly on the client machine using a package manager. Locked packages will also indirectly lock any dependent packages.

Procedure: Using Package Locks

1. On the client machine, install the `zypp-plugin-spacewalk` package:

```
# zypper in zypp-plugin-spacewalk
```

2. Navigate to the **Software › Packages › Lock** tab on the managed system to see a list of all available packages.
3. Select the packages to lock, and click **[Request Lock]**. You can also choose to enter a date and time for the lock to activate. Leave the date and time blank if you want the lock to activate as soon as possible. Note that the lock might not activate immediately.
4. To remove a package lock, select the packages to unlock and click **[Request Unlock]**. Leave the date and time blank if you want the lock to deactivate as soon as possible. Note that the lock might not deactivate immediately.

Registering Clients on a Proxy

Proxy servers can act as a broker and package cache for Salt clients. Registering clients on a Uyuni Proxy is very similar to registering them directly on Uyuni, with a few key differences.

This section contains information on registering Salt clients on a proxy using the Web UI, or with a bootstrap script.

Within the Web UI, standard proxy pages will show information about both Salt and traditional clients.

You can see a list of clients that are connected to a proxy by clicking on the name of the proxy in **Main Navigation › Systems › Systems › Proxy**, selecting the `Details` tab, and then selecting the `Proxy` tab.

A list of chained proxies for a Salt client can be seen by clicking on the name of the client in **Main Navigation › Systems › All**, selecting the `Details` tab, and then selecting the `Connection` tab.

If you decide to move any of your clients between proxies or the server you will need to repeat the registration process from the beginning.

Registering with the Web UI on a Proxy

Registering Salt clients to a Uyuni Proxy using the Web UI is similar to registering clients directly with the Uyuni Server.

Procedure: Registering clients to a proxy in the Web UI

1. In the Uyuni Web UI, navigate to **Systems › Bootstrapping**.
2. In the `Host` field, type the fully-qualified domain name (FQDN) of the client to be bootstrapped.
3. In the `SSH Port` field, type the SSH port number that will be used to connect and bootstrap the client. By default, the SSH port is `22`.
4. In the `User` field, type the username to log in to the client. By default, the username is `root`.
5. In the `Password` field, type password to log in to the client.
6. In the `Activation Key` field, select the activation key that is associated with the software channel you want to use to bootstrap the client.
7. In the `Proxy` field, select the proxy server you want to register to.
8. By default, the `Disable SSH Strict Key Host Checking` checkbox is selected. This allows the bootstrap process to automatically accept SSH host keys without requiring you to manually authenticate.
9. OPTIONAL: Check the `Manage System Completely via SSH` checkbox. If you check this option, the client will be configured to use SSH for its connection to the Server, and no other connection method will be configured.
10. Click **[Bootstrap]** to begin registration. When the bootstrap process has completed, your client will be listed at **Systems › System List**.

Instead of the Web UI, you may use the command line to register a minion through a proxy. This procedure requires that you have installed the Salt package on the minion before registration, and have the Advanced systems module activated.

Procedure: Register a Salt client through a proxy from the command line

1. Add the proxy FQDN as the master in the minions configuration file located at:

   ```
   /etc/salt/minion
   ```

   or alternatively:

   ```
   /etc/salt/minion.d/NAME.conf
   ```

2. Add the FQDN to the minion file:

   ```
   master: proxy123.example.com
   ```

   Save and restart the salt-minion service with:

   ```
   systemctl restart salt-minion
   ```

3. On the Server, accept the new minion key with:

   ```
   salt-key -a 'minion'
   ```

   The minion will now connect to the proxy exclusively for Salt operations and normal HTTP package downloads.

Registering with Bootstrap on a Proxy

Registering clients (either traditional or Salt) via SUSE Manager Proxy with a script is done almost the same way as registering clients directly with the Uyuni server. The difference is that you create the bootstrap script on the SUSE Manager Proxy with a command-line tool. The bootstrap script then deploys all necessary information to the clients. The bootstrap script requires some parameters (such as activation keys or GPG keys) that depend on your specific setup.

Procedure: Registering clients to a proxy with a bootstrap script

1. Create a client activation key on the Uyuni server using the Web UI. See Activation Keys.
2. On the proxy, execute the `mgr-bootstrap` command line tool as root. If needed, use the additional command line switches to tune your bootstrap script. To install a traditional client instead of a Salt client, ensure you use the `--traditional` switch.

To view available options type `mgr-bootstrap --help` from the command line:

```
# mgr-bootstrap --activation-keys=key-string
```

3. Optional: edit the resulting bootstrap script.
4. Execute the bootstrap script on the clients, following the instructions in run.bootstrap.script.

Introduction

Autoinstallation Types: AutoYaST and Kickstart

In the following section, AutoYaST and AutoYaST features apply for SUSE Linux Enterprise client systems only.

For RHEL systems, use Kickstart and Kickstart features.

Auto-Installing Salt Minions Currently Not Supported

This procedure will work for traditionally managed systems (system type `management`).

Autoinstallation is not currently available for systems using Salt (system type `salt`).

AutoYaST and Kickstart configuration files allow administrators to create an environment for automating otherwise time-consuming system installations, such as multiple servers or workstations. AutoYaST files have to be uploaded to be managed with Uyuni. Kickstart files can be created, modified, and managed within the Uyuni Web interface.

Uyuni also features the Cobbler installation server. For more information on Cobbler, see:

xref:FILENAME.adoc#advanced.topics.cobbler[].

AutoYaST

Using AutoYaST, a system administrator can create a single file containing the answers to all the questions that would normally be asked during a typical installation of a SUSE Linux Enterprise system.

AutoYaST files can be kept on a single server system and read by individual computers during the installation. This way the same AutoYaST file is used to install SUSE Linux Enterprise on multiple machines.

The *SUSE Linux Enterprise Server AutoYaST Guide* at (https://www.suse.com/documentation/sles-15/) will contain an in-depth discussion of “Automated Installation” using AutoYaST.

AutoYaST Explained

When a machine is to receive a network-based AutoYaST installation, the following events must occur in this order:

1. After being connected to the network and turned on, the machine’s PXE logic broadcasts its MAC address and requests to be discovered.
2. If no static IP address is used, the DHCP server recognizes the discovery request and offers network information needed for the new machine to boot. This includes an IP address, the default gateway to

be used, the netmask of the network, the IP address of the TFTP or HTTP server holding the bootloader program, and the full path and file name to that program (relative to the server's root).

3. The machine applies the networking information and initiates a session with the server to request the bootloader program.
4. The bootloader searches for its configuration file on the server from which it was loaded. This file dictates which Kernel and Kernel options, such as the initial RAM disk (initrd) image, should be executed on the booting machine. Assuming the bootloader program is SYSLINUX, this file is located in the `pxelinux.cfg` directory on the server and named the hexadecimal equivalent of the new machine's IP address. For example, a bootloader configuration file for SUSE Linux Enterprise Server should contain:

```
port 0
prompt 0
timeout 1
default autoyast
label autoyast
  kernel vmlinuz
  append autoyast=http://`my_susemanager_server`/`path`\
    install=http://`my_susemanager_server`/`repo_tree`
```

5. The machine accepts and uncompresses the initrd and kernel, boots the kernel, fetches the instsys from the install server and initiates the AutoYaST installation with the options supplied in the bootloader configuration file, including the server containing the AutoYaST configuration file.
6. The new machine is installed based on the parameters established within the AutoYaST configuration file.

AutoYaST Prerequisites

Some preparation is required for your infrastructure to handle AutoYaST installations. For instance, before creating AutoYaST profiles, you may consider:

- A DHCP server is not required for AutoYaST, but it can make things easier. If you are using static IP addresses, you should select static IP while developing your AutoYaST profile.
- Host the AutoYaST distribution trees via HTTP, properly provided by Uyuni.
- If conducting a so-called bare-metal AutoYaST installation, provide the following settings:
    - Configure DHCP to assign the required networking parameters and the bootloader program location.
    - In the bootloader configuration file, specify the kernel and appropriate kernel options to be used.

Building Bootable AutoYaST ISOs

While you can schedule a registered system to be installed by AutoYaST with a new operating system and package profile, you can also automatically install a system that is not registered with Uyuni, or does not yet have an operating system installed. One common method of doing this is to create a bootable CD-ROM that is inserted into the target system. When the system is rebooted or switched on, it boots from

the CD-ROM, loads the AutoYaST configuration from your Uyuni, and proceeds to install SUSE Linux Enterprise Server according to the AutoYaST profile you have created.

To use the CD-ROM, boot the system and type `autoyast` at the prompt (assuming you left the label for the AutoYaST boot as `autoyast`). When you press `Enter`, the AutoYaST installation begins.

For more information about image creation, refer to KIWI at http://doc.opensuse.org/projects/kiwi/doc/.

Integrating AutoYaST with PXE

In addition to CD-ROM-based installations, AutoYaST installation through a Pre-Boot Execution Environment (PXE) is supported. This is less error-prone than CDs, enables AutoYaST installation from bare metal, and integrates with existing PXE/DHCP environments.

To use this method, make sure your systems have network interface cards (NIC) that support PXE, install and configure a PXE server, ensure DHCP is running, and place the installation repository on an HTTP server for deployment. Finally upload the AutoYaST profile via the Web interface to the Uyuni server. Once the AutoYaST profile has been created, use the URL from the **Autoinstallation Overview** page, as for CD-ROM-based installations.

To obtain specific instructions for conducting PXE AutoYaST installation, refer to the *Using PXE Boot* section of the *SUSE Linux Enterprise Deployment Guide*.

Starting with xref:FILENAME.adoc#ref.webui.systems.autoinst.profiles[], AutoYaST options available from **Systems › Kickstart** are described.

Kickstart

Using Kickstart, a system administrator can create a single file containing the answers to all the questions that would normally be asked during a typical installation of Red Hat Enterprise Linux.

Kickstart files can be kept on a single server and read by individual computers during the installation. This method allows you to use one Kickstart file to install Red Hat Enterprise Linux on multiple machines.

The *Red Hat Enterprise Linux System Administration Guide* contains an in-depth description of Kickstart (https://access.redhat.com/documentation/en/red-hat-enterprise-linux/).

Kickstart Explained

When a machine is to receive a network-based Kickstart, the following events must occur in this order:

1. After being connected to the network and turned on, the machine's PXE logic broadcasts its MAC address and requests to be discovered.
2. If no static IP address is used, the DHCP server recognizes the discovery request and offers network information needed for the new machine to boot. This information includes an IP address, the default gateway to be used, the netmask of the network, the IP address of the TFTP or HTTP server holding the bootloader program, and the full path and file name of that program (relative to the server's

root).

3. The machine applies the networking information and initiates a session with the server to request the bootloader program.
4. The bootloader searches for its configuration file on the server from which it was loaded. This file dictates which kernel and kernel options, such as the initial RAM disk (initrd) image, should be executed on the booting machine. Assuming the bootloader program is SYSLINUX, this file is located in the `pxelinux.cfg` directory on the server and named the hexadecimal equivalent of the new machine's IP address. For example, a bootloader configuration file for Red Hat Enterprise Linux AS 2.1 should contain:

```
port 0
prompt 0
timeout 1
default My_Label
label My_Label
      kernel vmlinuz
      append ks=http://`my_susemanager_server`/`path`\
          initrd=initrd.img network apic
```

5. The machine accepts and uncompresses the init image and kernel, boots the kernel, and initiates a Kickstart installation with the options supplied in the bootloader configuration file, including the server containing the Kickstart configuration file.
6. This Kickstart configuration file in turn directs the machine to the location of the installation files.
7. The new machine is built based on the parameters established within the Kickstart configuration file.

Kickstart Prerequisites

Some preparation is required for your infrastructure to handle Kickstarts. For instance, before creating Kickstart profiles, you may consider:

- A DHCP server is not required for kickstarting, but it can make things easier. If you are using static IP addresses, select static IP while developing your Kickstart profile.
- An FTP server can be used instead of hosting the Kickstart distribution trees via HTTP.
- If conducting a bare metal Kickstart, you should configure DHCP to assign required networking parameters and the bootloader program location. Also, specify within the bootloader configuration file the kernel to be used and appropriate kernel options.

Building Bootable Kickstart ISOs

While you can schedule a registered system to be kickstarted to a new operating system and package profile, you can also Kickstart a system that is not registered with Uyuni or does not yet have an operating system installed. One common method of doing this is to create a bootable CD-ROM that is inserted into the target system. When the system is rebooted, it boots from the CD-ROM, loads the Kickstart configuration from your Uyuni, and proceeds to install Red Hat Enterprise Linux according to the Kickstart profile you have created.

To do this, copy the contents of `/isolinux` from the first CD-ROM of the target distribution. Then edit the `isolinux.cfg` file to default to 'ks'. Change the 'ks' section to the following template:

```
label ks
kernel vmlinuz
  append text ks=`url`initrd=initrd.img lang= devfs=nomount \
    ramdisk_size=16438`ksdevice`
```

IP address-based Kickstart URLs will look like this:

```
http://`my.manager.server`/kickstart/ks/mode/ip_range
```

The Kickstart distribution defined via the IP range should match the distribution from which you are building, or errors will occur. `ksdevice` is optional, but looks like:

```
ksdevice=eth0
```

It is possible to change the distribution for a Kickstart profile within a family, such as Red Hat Enterprise Linux AS 4 to Red Hat Enterprise Linux ES 4, by specifying the new distribution label. Note that you cannot move between versions (4 to 5) or between updates (U1 to U2).

Next, customize `isolinux.cfg` further for your needs by adding multiple Kickstart options, different boot messages, shorter timeout periods, etc.

Next, create the ISO as described in the *Making an Installation Boot CD-ROM* section of the *Red Hat Enterprise Linux Installation Guide*. Alternatively, issue the command:

```
mkisofs -o file.iso -b isolinux.bin -c boot.cat -no-emul-boot \
  -boot-load-size 4 -boot-info-table -R -J -v -T isolinux/
```

Note that `isolinux/` is the relative path to the directory containing the modified isolinux files copied from the distribution CD, while `file.iso` is the output ISO file, which is placed into the current directory.

Burn the ISO to CD-ROM and insert the disc. Boot the system and type "ks" at the prompt (assuming you left the label for the Kickstart boot as 'ks'). When you press `Enter`, Kickstart starts running.

Integrating Kickstart with PXE

In addition to CD-ROM-based installs, Kickstart supports a Pre-Boot Execution Environment (PXE). This is less error-prone than CDs, enables kickstarting from bare metal, and integrates with existing PXE/DHCP environments.

To use this method, make sure your systems have network interface cards (NIC) that support PXE. Install and configure a PXE server and ensure DHCP is running. Then place the appropriate files on an HTTP

server for deployment. Once the Kickstart profile has been created, use the URL from the **Kickstart Details** page, as for CD-ROM-based installs.

To obtain specific instructions for conducting PXE Kickstarts, refer to the *PXE Network Installations* chapter of the *Red Hat Enterprise Linux 4 System Administration Guide*.

Running the Network Booting Tool, as described in the Red Hat Enterprise Linux 4: System Administration Guide, select "HTTP" as the protocol and include the domain name of the Uyuni in the Server field if you intend to use it to distribute the installation files.

The following sections describe the autoinstallation options available from the **Systems › Autoinstallation** page.

Cobbler

Introduction

Uyuni features the Cobbler server, which allows administrators to centralize system installation and provisioning infrastructure. Cobbler is an installation server that provides various methods of performing unattended system installations. It can be used on server, workstation, or guest systems, in full or para-virtualized environments.

Cobbler offers several tools for pre-installation guidance, automated installation file management, installation environment management, and more. This section explains some of the supported features of Cobbler, including:

- Installation environment analysis using the `cobbler check` command
- Multi-site installation server configuration using the `cobbler replicate` command
- Virtual machine guest installation automation with the `koan` client-side tool
- Building installation ISOs with PXE-like menus using the `cobbler buildiso` command (for Uyuni systems with x86_64 architecture)

For more detailed Cobbler documentation, see http://cobbler.github.io/manuals/.

Supported Cobbler Functions

SUSE only support those Cobbler functions that are either listed within our formal documentation or available via the web interface and API.

Cobbler Requirements

To use Cobbler for system installation with PXE, you will require a TFTP server. Uyuni installs a TFTP server by default. To PXE boot systems, you will require a DHCP server, or have access to a network DHCP server. Edit the `/etc/dhcp.conf` configuration file to change `next-server` to the hostname

or IP address of your Cobbler server.

Cobbler requires an open HTTP port to synchronize data between the Server and the Proxy. By default, Cobbler uses port 80, but you can configure it to use port 443 instead if that suits your environment.

Correct Hostname Configuration

Cobbler uses hostnames as a unique key for each system. If you are using the `pxe-default-image` to onboard bare metal systems, make sure every system has a unique hostname. Non-unique hostnames will cause all systems with the same hostname to have the configuration filess overwritten when a provisioning profile is assigned.

Configuring Cobbler with /etc/cobbler/settings

Cobbler configuration is primarily managed using the `/etc/cobbler/settings` file. Cobbler will run with the default settings unchanged. All configurable settings are explained in detail in the `/etc/cobbler/settings` file, including information on each setting, and recommendations.

Supported Languages

If Uyuni complains that language `en` was not found within the list of supported languages available at `/usr/share/YaST2/data/languages`, remove the `lang` parameter in the `/etc/cobbler/settings` file, or add a valid parameter such as `en_US`.

For more on this topic, see https://www.suse.com/support/kb/doc?id=7018334.

Cobbler and DHCP

Cobbler uses DHCP to automate bare metal installations from a PXE boot server. You must have administrative access to the network's DHCP server, or be able to configure DHCP directly on the the Cobbler server.

Configuring an Existing DHCP Server

If you have existing DHCP server, you will need to edit the DHCP configuration file so that it points to the Cobbler server and PXE boot image.

As root on the DHCP server, edit the `/etc/dhcpd.conf` file and append a new class with options for performing PXE boot installation. For example:

```
allow booting;
allow bootp; ①
class "PXE" ②
{match if substring(option vendor-class-identifier, 0, 9) = "PXEClient"; ③
next-server 192.168.2.1; ④
filename "pxelinux.0";} ⑤
```

① Enable network booting with the `bootp` protocol.

② Create a class called `PXE`.

③ A system configured to have PXE first in its boot priority identifies itself as `PXEClient`.

④ As a result, the DHCP server directs the system to the Cobbler server at `192.168.2.1`.

⑤ The DHCP server retrieves the `pxelinux.0`bootloader file.

Setting up PXE Boot in KVM

It is possible to set up PXE booting in KVM, however we do not recommend you use this method for production systems. This method can replace the `next-server` setting on a DHCP server, as described in xref:FILENAME.adoc#advanced.topics.cobbler.reqs.dhcp.notmanaged[]. Edit the network XML description with `virsh`:

1. Produce an XML dump of the current description:

```
virsh net-dumpxml --inactive network > network.xml
```

2. Open the XML dump file at `network.xml` with a text editor and add a `bootp` parameter within the `<dhcp>` element:

```
<bootp file='/pxelinux.0' server='192.168.100.153'/>
```

3. Install the updated description:

```
virsh net-define network.xml
```

Alternatively, use the `net-edit` subcommand, which will also perform some error checking.

Example 1. Minimal Network XML Description for KVM

```
<network>
  <name>default</name>
  <uuid>1da84185-31b5-4c8b-9ee2-a7f5ba39a7ee</uuid>
  <forward mode='nat'>
    <nat>
      <port start='1024' end='65535'/>
    </nat>
  </forward>
  <bridge name='virbr0' stp='on' delay='0'/>
  <mac address='52:54:00:29:59:18'/>
  <domain name='default'/>
  <ip address='192.168.100.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.100.128' end='192.168.100.254'/>
      <bootp file='/pxelinux.0' server='192.168.100.153'/> ①
</dhcp>
  </ip>
</network>
```

① `bootp` statement that directs to the PXE server.

TFTP

Uyuni uses the `atftpd` daemon, but it can also use TFTP. The `atftpd` daemon is the recommended method for PXE serviices, and is installed by default. Usually, you do not have to change its configuration, but if you have to, use the YaST Services Manager.

Before TFTP can serve the `pxelinux.0` boot image, you must start the tftp service. Start YaST and use **System › Services Manager** to configure the `tftpd` daemon.

Syncing TFTP Contents to Uyuni Proxies

It is possible to synchronize Cobbler-generated TFTP contents to Uyuni proxies to perform PXE booting using proxies.

Installation

On the Uyuni Server as the root user, install the `susemanager-tftpsync` package:

```
zypper install susemanager-tftpsync
```

On the SUSE Manager Proxy systems as the root user , install the `susemanager-tftpsync-recv` package:

```
zypper install susemanager-tftpsync-recv
```

Configuring SUSE Manager Proxy

Execute `configure-tftpsync.sh` on the SUSE Manager Proxy systems.

This setup script asks for hostnames and IP addresses of the Uyuni server and the proxy. Additionally, it asks for the `tftpboot` directory on the proxy. For more information, see the output of `configure-tftpsync.sh --help`.

Configuring Uyuni Server

As the root user, execute `configure-tftpsync.sh` on Uyuni Server:

```
configure-tftpsync.sh proxy1.example.com proxy2.example.com
```

Execute `cobbler sync` to initially push the files to the proxy systems. This will succeed if all listed proxies are properly configured.

Changing the List of Proxy Systems

You can call `configure-tftpsync.sh` to change the list of proxy systems. You must always provide the full list of proxy systems.

Reinstalling a Configured Proxy

If you reinstall an already configured proxy and want to push all the files again you must remove the cache file at `/var/lib/cobbler/pxe_cache.json` before you can call `cobbler sync` again.

Requirements

The Uyuni Server must be able to access the SUSE Manager Proxy systems directly. You cannot push using a proxy.

Syncing and Starting the Cobbler Service

Before starting the Cobbler service, run a check to make sure that all the prerequisites are configured according to your requirements using the `cobbler check` command.

If configuration is correct, start the Uyuni server with this command:

```
/usr/sbin/spacewalk-service start
```

Do not start or stop the `cobblerd` service independent of the Uyuni service. Doing so may cause errors and other issues.

Always use `/usr/sbin/spacewalk-service` to start or stop Uyuni.

Adding a Distribution to Cobbler

If all Cobbler prerequisites have been met and Cobbler is running, you can use the Cobbler server as an installation source for a distribution:

Make installation files such as the kernel image and the initrd image available on the Cobbler server. Then add a distribution to Cobbler, using either the Web interface or the command line tools.

For information about creating and configuring AutoYaST or Kickstart distributions from the Uyuni interface, refer to xref:FILENAME.adoc#ref.webui.systems.autoinst.distribution[].

To create a distribution from the command line, use the `cobbler` command as root:

```
cobbler distro add --name=`string`--kernel=`path`--initrd=`path`
```

`--name=string` option

A label used to differentiate one distribution choice from another (for example, `sles12server`).

`--kernel=path` option

Specifies the path to the kernel image file.

`--initrd=path` option

specifies the path to the initial ram disk (initrd) image file.

Adding a Profile to Cobbler

Once you have added a distribution to Cobbler, you can add profiles.

Cobbler profiles associate a distribution with additional options like AutoYaST or Kickstart files. Profiles are the core unit of provisioning and there must be at least one Cobbler profile for every distribution added. For example, two profiles might be created for a Web server and a desktop configuration. While both profiles use the same distribution, the profiles are for different installation types.

For information about creating and configuring Kickstart and AutoYaST profiles in the Uyuni interface, refer to xref:FILENAME.adoc#ref.webui.systems.autoinst.profiles[].

Use the `cobbler` command as root to create profiles from the command line:

```
cobbler profile add --name=string --distro=string [--kickstart=url] \
  [--virt-file-size=gigabytes] [--virt-ram=megabytes]
```

`--name=string`

A unique label for the profile, such as `sles12webserver` or `sles12workstation`.

`--distro=string`

The distribution that will be used for this profile. For adding distributions, see xref:FILENAME.adoc#advanced.topics.cobbler.adddistro[].

`--kickstart=url`

The location of the Kickstart file (if available).

`--virt-file-size=gigabytes`

The size of the virtual guest file image (in gigabytes). The default is 5 GB.

`--virt-ram=megabytes`

The maximum amount of physical RAM a virtual guest can consume (in megabytes). The default is 512 MB.

Adding a System to Cobbler

Once the distributions and profiles for Cobbler have been created, add systems to Cobbler. System records map a piece of hardware on a client with the Cobbler profile assigned to run on it.

If you are provisioning using `koan` and PXE menus alone, it is not required to create system records. They are useful when system-specific Kickstart templating is required or to establish that a specific system should always get specific content installed. If a client is intended for a certain role, system records should be created for it.

For information about creating and configuring automated installation from the Uyuni interface, refer to xref:FILENAME.adoc#s4-sm-system-details-kick[].

Run this command as the root user to add a system to the Cobbler configuration:

```
cobbler system add --name=string --profile=string \
  --mac-address=AA:BB:CC:DD:EE:FF
```

`--name=string`

A unique label for the system, such as `engineering_server` or `frontoffice_workstation`.

`--profile=string`

Specifies the name of one of the profiles added in xref:FILENAME.adoc#advanced.topics.cobbler.addprofile[].

`--mac-address=AA:BB:CC:DD:EE:FF`

Allows systems with the specified MAC address to automatically be provisioned to the profile associated with the system record when they are being installed.

For more options, such as setting hostname or IP addresses, refer to the Cobbler manpage (`man cobbler`).

Using Cobbler Templates

The Uyuni web interface allows you to create variables for use with Kickstart distributions and profiles. For more information on creating Kickstart profile variables, refer to xref:FILENAME.adoc#s4-sm-system-kick-details-variables[].

Kickstart variables are part of an infrastructure change in Uyuni to support templating in Kickstart files. Kickstart templates are files that describe how to build Kickstart files, rather than creating specific Kickstarts. The templates are shared by various profiles and systems that have their own variables and corresponding values. These variables modify the templates and a template engine parses the template and variable data into a usable Kickstart file. Cobbler uses an advanced template engine called Cheetah that provides support for templates, variables, and snippets.

Advantages of using templates include:

- Robust features that allow administrators to create and manage large amounts of profiles or systems without duplication of effort or manually creating Kickstarts for every unique situation.
- While templates can become complex and involve loops, conditionals and other enhanced features and syntax, you can also create simpler Kickstart files without such complexity.

Using Templates

Kickstart templates can have static values for certain common items such as PXE image file names, subnet addresses, and common paths such as `/etc/sysconfig/network-scripts/`. However, templates differ from standard Kickstart files in their use of variables.

For example, a standard Kickstart file may have a networking section similar to this:

```
network --device=eth0 --bootproto=static --ip=192.168.100.24 \
  --netmask=255.255.255.0 --gateway=192.168.100.1 --nameserver=192.168.100.2
```

In a Kickstart template file, the networking section would look like this instead:

```
network --device=$net_dev --bootproto=static --ip=$ip_addr \
  --netmask=255.255.255.0 --gateway=$my_gateway --nameserver=$my_nameserver
```

These variables are substituted with the values set in your Kickstart profile variables or in your system detail variables. If the same variable is defined in both the profile and the system detail, then the system detail variable takes precedence.

The template for the autoinstallation has syntax rules which relies on punctuation symbols. To avoid clashes, they need to be properly treated.

In case the autoinstallation scenario contains any shell script using variables like `$(example)`, its content should be escaped by using the backslash symbol: `\$(example)`.

If the variable named `example` is defined in the autoinstallation snippet, the templating engine will evaluate `$example` with its content. If there is no such variable, the content will be left unchanged. Escaping the `$` symbol will prevent the templating engine from evaluating the symbol as an internal variable. Long scripts or strings can be escaped by wrapping them with the `\#raw` and `\#end raw` directives. For example:

```
#raw
#!/bin/bash
for i in {0..2}; do
 echo "$i - Hello World!"
done
#end raw
```

Also, pay attention to scenarios like this:

```
#start some section (this is a comment)
echo "Hello, world"
#end some section (this is a comment)
```

Any line with a `#` symbol followed by a whitespace is treated as a comment and is therefore not evaluated.

For more information about Kickstart templates, refer to the Cobbler project page at:

https://fedorahosted.org/cobbler/wiki/KickstartTemplating

Kickstart Snippets

If you have common configurations across all Kickstart templates and profiles, you can use the Snippets feature of Cobbler to take advantage of code reuse.

Kickstart snippets are sections of Kickstart code that can be called by a `$SNIPPET()` function that will be parsed by Cobbler and substituted with the contents of the snippet.

For example, you might have a common hard drive partition configuration for all servers, such as:

```
clearpart --all
part /boot --fstype ext3 --size=150 --asprimary
part / --fstype ext3 --size=40000 --asprimary
part swap --recommended

part pv.00 --size=1 --grow

volgroup vg00 pv.00
logvol /var --name=var vgname=vg00 --fstype ext3 --size=5000
```

Save this snippet of the configuration to a file like `my_partition` and place the file in

`/var/lib/cobbler/snippets/`, where Cobbler can access it.

Use the snippet by calling the `$SNIPPET()` function in your Kickstart templates. For example:

```
$SNIPPET('my_partition')
```

Wherever you invoke that function, the Cheetah parser will substitute the function with the snippet of code contained in the `my_partition` file.

Using Koan

Whether you are provisioning guests on a virtual machine or reinstalling a new distribution on a running system, Koan works in conjunction with Cobbler to provision systems.

Using Koan to Provision Virtual Systems

If you have created a virtual machine profile as documented in xref:FILENAME.adoc#advanced.topics.cobbler.addprofile[], you can use `koan` to initiate the installation of a virtual guest on a system. For example, create a Cobbler profile with the following command:

```
cobbler add profile --name=virtualfileserver \
  --distro=sles12-x86_64-server --virt-file-size=20 --virt-ram=1000
```

This profile is for a fileserver running SUSE Linux Enterprise Server 12 with a 20 GB guest image size and allocated 1 GB of system RAM. To find the name of the virtual guest system profile, use the `koan` command:

```
koan --server=hostname --list-profiles
```

This command lists all the available profiles created with `cobbler profile add`.

Create the image file, and begin installation of the virtual guest system:

```
koan --virt --server=cobbler-server.example.com \
  --profile=virtualfileserver --virtname=marketingfileserver
```

This command specifies that a virtual guest system be created from the Cobbler server (hostname `cobbler-server.example.com`) using the `virtualfileserver` profile. The `virtname` option specifies a label for the virtual guest, which by default is the system's MAC address.

Once the installation of the virtual guest is complete, it can be used as any other virtual guest system.

Using Koan to Reinstall Running Systems

`koan` can replace a still running system with a new installation from the available Cobbler profiles by executing the following command *on the system to be reinstalled*:

```
koan --replace-self --server=hostname --profile=name
```

This command, running on the system to be replaced, will start the provisioning process and replace the system with the profile in `--profile=name` on the Cobbler server specified in `--server=hostname`.

Building ISOs with Cobbler

Some environments might lack PXE support. The Cobbler `buildiso` command creates a ISO boot image containing a set of distributions and kernels, and a menu similar to PXE network installations. Define the name and output location of the boot ISO using the `--iso` option.

ISO Build Directory

Depending on Cobbler-related systemd settings (see `/usr/lib/systemd/system/cobblerd.service`) writing ISO images to public `tmp` directories will not work.

```
cobbler buildiso --iso=/path/to/boot.iso
```

The boot ISO includes all profiles and systems by default. Limit these profiles and systems using the `--profiles` and `--systems` options.

```
cobbler buildiso --systems="system1,system2,system3" \
  --profiles="profile1,profile2,profile3"
```

Building ISOs with the `cobbler buildiso` command is supported for all architectures except the IBM Z architecture.

Bare Metal Provisioning

Systems that have not yet been provisioned are called bare metal systems. You can provision bare metal systems using Cobbler. Once a bare metal system has been provisioned in this way, it will appear in the `Systems` list, where you can perform regular provisioning with autoinstallation, for a completely unattended installation.

Bare Metal Provisioning System Requirements

To successfully provision a bare metal system, you will require a fully patched Uyuni server, version 2.1 or higher.

The system to be provisioned must have x86_64 architecture, with at least 2 GB RAM, and be capable of PXE booting.

The server uses TFTP to provision the bare metal client, so the appropriate port and networks must be configured correctly in order for provisioning to be successful. In particular, ensure that you have a DHCP server, and have set the `next-server` parameter to the Uyuni server IP address or hostname.

Enabling Bare Metal Systems Management

Bare metal systems management can be enabled or disabled in the Web UI by clicking **Admin › SUSE Manager Configuration › Bare-metal systems**.

New systems are added to the organization of the administrator who enabled the bare metal systems management feature. To change the organization, log in as an Administrator of the required organization, and re-enable the feature.

Once the feature has been enabled, any bare metal system connected to the server network will be automatically added to the organization when it is powered on. The process can take a few minutes, and the system will automatically shut down once it is complete. After the reboot, the system will appear in the `Systems` list. Click on the name of the system to see basic information, or go to the `Properties`, `Notes`, and `Hardware` tabs for more details. You can migrate bare metal systems to other organizations if required, using the `Migrate` tab.

Provisioning Bare Metal Systems

Provisioning bare metal systems is similar to provisioning other systems, and can be done using the `Provisioning` tab. However, you will not be able to schedule provisioning, it will happen automatically as soon as the system is configured and powered on.

Bare Metal and System Set Manager

System Set Manager can be used with bare metal systems, although not all features will be available, because bare metal systems do not have an operating system installed. This limitation also applies to mixed sets that contain bare metal systems; all features will be re-enabled if the bare metal systems are removed from the set.

Troubleshooting Bare Metal Systems

If a bare metal system on the network is not automatically added to the `Systems` list, check these things first:

- You must have the `pxe-default-image` package installed.
- File paths and parameters must be configured correctly. Check that the `vmlinuz0` and `initrd0.img` files, which are provided by `pxe-default-image`, are in the locations specified in the `rhn.conf` configuration file.

- Ensure the networking equipment connecting the bare metal system to the Uyuni server is working correctly, and that you can reach the Uyuni server IP address from the server.
- The bare metal system to be provisioned must have PXE booting enabled in the boot sequence, and must not be attempting to boot an operating system.
- The DHCP server must be responding to DHCP requests during boot. Check the PXE boot messages to ensure that:
  - the DHCP server is assigning the expected IP address
  - the DHCP server is assigning the the Uyuni server IP address as `next-server` for booting.
- Ensure Cobbler is running, and that the Discovery feature is enabled.

If you see a blue Cobbler menu shortly after booting, discovery has started. If it does not complete successfully, temporarily disable automatic shutdown in order to help diagnose the problem. To disable automatic shutdown:

1. Select `pxe-default-profile` in the Cobbler menu with the arrow keys, and press the Tab key before the timer expires.
2. Add the kernel boot parameter `spacewalk-finally=running` using the integrated editor, and press Enter to continue booting.
3. Enter a shell with the username `root` and password `linux` to continue debugging.

Duplicate profiles

Due to a technical limitation, it is not possible to reliably distinguish a new bare metal system from a system that has previously been discovered. Therefore, we recommended that you do not power on bare metal systems multiple times, as this will result in duplicate profiles.

Disconnected Setup with RMT or SMT (DMZ)

When it is not possible to connect Uyuni directly or via a proxy to the Internet, a disconnected setup in combination with RMT or SMT is the recommended solution.

In this scenario, RMT or SMT stays in an “external” network with a connection to SUSE Customer Center and synchronizes the software channels and repositories on a removable storage medium. Then you can separate the storage medium from RMT or SMT, and mount it locally on your Uyuni server to read the updated data.

Offline Usage Scenario

SMT and RMT are not made for server cascades. SUSE Manager always connects to SMT or RMT in an offline or disconnected scenario.

RMT

The successor of SMT and currently runs on the following systems:

- SUSE Linux Enterprise 15 (when available)
- Temporarily (for testing only): 12 SP2, and 12 SP3
- Not officially supported: openSUSE Leap 42.2, Leap 42.3, and openSUSE Tumbleweed

RMT allows you to provision updates for all of your devices running a product based on SUSE Linux Enterprise 12 SPx and later as well as openSUSE Leap.

SMT

The predecessor of RMT and is no longer actively developed. It runs on SUSE Linux Enterprise Server 12 SPx and allows you to provision updates for products based on SUSE Linux Enterprise 12 SPx and earlier. You will still need it, if you want to update SUSE Linux Enterprise 11 clients.

Repository Management Tool (RMT) and Disconnected Setup (DMZ)

The following procedure will guide you through using RMT. It will work best with a dedicated RMT instance per Uyuni .

Procedure: RMT: Fetching Repository Data from SUSE Customer Center

1. Configure RMT in the external network with SCC. For details about configuring RMT, see the official guide (when available).
    a. Preparation work:

    Run `rmt-cli sync` to download available products and repositories data for your organization from SCC.

    Run `rmt-cli products list --all` to see the list of products that are available for your organization.

    Run `rmt-cli repos list --all` to see the list of all repositories available.

    b. With `rmt-cli repos enable` enable repositories you want to mirror.

    c. With `rmt-cli products enable`enable products. For example, to enable SLES _15:

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

2. Using RMT, mirror all required repositories.
3. Get the required JSON responses from SCC and save them as files at the specified path (for example, `/mnt/usb`).

Write Permissions for RMT User

The directory being written to must be writeable for the same user as the rmt service. The rmt user setting is defined in the `cli` section of `/etc/rmt.conf` .

Enter:

```
{prompt.root}rmt-cli export data /mnt/usb
```

4. Export settings about repositories to mirror to the specified path (in this case, `/mnt/usb`); this command will create a `repos.json` file there:

```
{prompt.root}rmt-cli export settings /mnt/usb
```

5. Mirror the repositories according to the settings in the `repos.json` file to the specified path (in this case, `/mnt/usb`).

```
{prompt.root}rmt-cli export repos /mnt/usb
```

6. Unmount the storage medium and carry it securely to your Uyuni server.

On the Uyuni server, continue with Updating Repositories on Uyuni From Storage Media.

Repository Management Tool (SMT) and Disconnected Setup (DMZ)

The following procedure will guide you through using SMT.

Procedure: SMT: Fetching Repository Data from SUSE Customer Center

1. Configure SMT in the external network with SCC. For details about configuring SMT with SUSE Linux Enterprise 12, see https://www.suse.com/documentation/sles-12/book_smt/data/book_smt.html.
2. Using SMT, mirror all required repositories.
3. Create a “database replacement file” (for example, `/tmp/dbrepl.xml`).

```
{prompt.root}smt-sync --createdbreplacementfile /tmp/dbrepl.xml
```

1. Mount a removable storage medium such as an external hard disk or USB flash drive.
2. Export the data to the mounted medium:

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

Write Permissions for SMT User

The directory being written to must be writeable for the same user as the smt daemon (user=smt). The smt user setting is defined in `/etc/smt.conf` . You can check if the correct user is specified via the following command:

```
{prompt.root}egrep '^smtUser' /etc/smt.conf
```

+

+ .Keeping the Disconnected Server Up-to-date NOTE: `smt-sync` also exports your subscription data. To keep Uyuni up-to-date with your subscriptions, you must frequently import and export this data.

+

1. Unmount the storage medium and carry it securely to your Uyuni server.

On the Uyuni server, continue with Updating Repositories on Uyuni From Storage Media.

Updating Repositories on Uyuni From Storage Media

This procedure will show you how to update the repositories on the Uyuni server from the storage media.

Procedure: Updating the UyuniServer from the Storage Medium

1. Mount the storage medium on your Uyuni server (for example, at `/media/disk`).
2. Specify the local path on the Uyuni server in `/etc/rhn/rhn.conf`:

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

   This setting is mandatory for SUSE Customer Center and `mgr-sync`.
3. Restart Tomcat:

   ```
   systemctl restart tomcat
   ```

1. Before performing another operation on the server execute a full sync:

   ```
   mgr-sync refresh   # SCC (fromdir in rhn.conf required!)
   ```

2. `mgr-sync` can now be executed normally:

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

Data Corruption

The disk must always be available at the same mount point. To avoid data corruption, do not trigger a sync, if the storage medium is not mounted. If you have already added a channel from a local repository path, you will not be able to change its URL to point to a different path afterwards.

Up-to-date data is now available on your Uyuni server and is ready for updating client systems. According to your maintenance windows or update schedule refresh the data on the storage medium with RMT or SMT.

Refreshing Data on the Storage Medium

Procedure: Refreshing Data on the Storage Medium from RMT or SMT

1. On your Uyuni server, unmount the storage medium and carry it to your RMT or SMT.
2. On your RMT or SMT system, continue with the synchronization step.

Data Corruption

The storage medium must always be available at the same mount point. To avoid data corruption, do not trigger a sync if the storage medium is not mounted.

This concludes using RMT or SMT with Uyuni .

Other Clients

It is possible to register clients using operating systems from Red Hat, CentOS, or Ubuntu.

This section contains information specific to clients running operating systems other than those provided by SUSE.

Managing Red Hat Enterprise Linux Clients

The following sections provide guidance on managing Red Hat Expanded Support clients, this includes Salt minions and traditional systems.

Server Configuration for Red Hat Enterprise Linux Channels

This section provides guidance on server configuration for Red Hat Enterprise Linux Channels provided by SUSE.

- Minimum of 8 GB RAM and at least two physical or virtual CPUs. Taskomatic will use one of these CPUs.
- Taskomatic requires of minimum of 3072 MB RAM. This should be set in `/etc/rhn/rhn.conf`:

```
taskomatic.java.maxmemory=3072
```

- Provision enough disk space. `/var/spacewalk` contains all mirrored RPMs. For example, Red Hat Enterprise Linux 6 x86_64 channels require 90 GB and more.
- LVM or an NFS mount is recommended.
- Access to RHEL 5/6/7 Subscription Media.

Access to RHEL Media or Repositories

Access to Red Hat base media repositories and RHEL installation media is the responsibility of the user. Ensure that all your RHEL systems obtain support from RHEL or all your RHEL systems obtain support from SUSE. If you do not follow these practices you may violate terms with Red Hat.

Red Hat Enterprise Linux Channel Management Tips

This section provides tips on Red Hat Enterprise Linux channel management.

- The base parent distribution Red Hat Enterprise Linux channel per architecture contains zero packages. No base media is provided by SUSE. The RHEL media or installation ISOs should be added as child channels of the Red Hat Enterprise Linux parent channel.
- The Red Hat Enterprise Linux and tools channels are provided by SUSE Customer Center (SCC) using `mgr-sync`.

- It can take up to 24 hours for an initial channel synchronization to complete.
- When you have completed the initial synchronization process of any Red Hat Enterprise Linux channel it is recommended to clone the channel before working with it. This provides you with a backup of the original synchronization.

Mirroring RHEL Media into a Channel

The following procedure guides you through setup of the RHEL media as a Uyuni channel. All packages on the RHEL media will be mirrored into a child channel located under RES 5/6/7 distribution per architecture.

Procedure: Mirroring RHEL Media into a Channel

1. Create a new Channel by log in to the Web UI and selecting **Channels › Manage Software Channels › Create Channel** .
2. Fill in basic channel details and add the channel as a child to the corresponding RES 5/6/7 distribution channel per architecture from SCC. The base parent channel should contain zero packages.
3. Modify the RES 5/6/7 activation key to include this new child channel.
4. As root on the Uyuni command line copy the ISO to the `/tmp` directory.
5. Create a directory to contain the media content:

```
{prompt.root}mkdir -p /srv/www/htdocs/pub/rhel
```

6. Mount the ISO:

```
{prompt.root}mount -o loop /tmp/name_of_iso /srv/www/htdocs/pub/rhel
```

7. Start `spacewalk-repo-sync` to synchronize Red Hat Enterprise Linux 7 packages:

```
{prompt.root}spacewalk-repo-sync -c channel_name -u https://127.0.0.1/pub/rhel/
Repo URL: https://127.0.0.1/pub/rhel/
Packages in repo:              [...]
Packages already synced:       [...]
Packages to sync:              [...]
[...]
```

   To synchronize RES 5/6 packages:

```
{prompt.root}spacewalk-repo-sync -c channel_name -u https://127.0.0.1/pub/rhel/Server/
Repo URL: https://127.0.0.1/pub/rhel/Server/
Packages in repo:              [...]
Packages already synced:       [...]
Packages to sync:              [...]
[...]
```

8. When the channel has completed the synchronization process you can use the channel as any normal Uyuni channel.

Attempting to synchronize the repository will sometimes fail with this error:

```
[Errno 256] No more mirrors to try.
```

To troubleshoot this error, look at the HTTP protocol to determine if `spacewalk-repo-sync` is running:

procedure: Debug spacewalk-repo-sync

1. Start debugging mode with `export URLGRABBER_DEBUG=DEBUG`
2. Check the output of `/usr/bin/spacewalk-repo-sync --channel <channel-label> --type yum`
3. If you want to disable debug mode, use `unset URLGRABBER_DEBUG`

Registering RES Salt Minions with Uyuni

This section will guide you through registering RHEL minions with Uyuni.

This section assumes you have updated your server to the latest patch level.

Synchronizing Appropriate Red Hat Enterprise Linux Channels

Ensure you have the corresponding Red Hat Enterprise Linux product enabled and required channels have been fully synchronized:

RHEL 7.x

- Product: Red Hat Enterprise Linux 7
- Mandatory channels: `rhel-x86_64-server-7` , `res7-suse-manager-tools-x86_64` , `res7-x86_64` systemitem>

RHEL 6.x

- Product: Red Hat Enterprise Linux 6
- Mandatory channels: `rhel-x86_64-server-6` , `res6-suse-manager-tools-x86_64` , `res6-x86_64`

Checking Synchronization Progress

To check if a channel has finished synchronizing you can do one of the following:

- From the UyuniWeb UI browse to **Admin** › **Setup Wizard** and select the **SUSE Products** tab. Here you will find a percent completion bar for each product.
- Alternatively, you may check the synchronization log file located under `/var/log/rhn/reposync/channel-label.log` using cat or the tailf command. Keep in mind that base channels can contain multiple child channels. Each of these child channels will generate its own log during the synchronization progress. Do not assume a channel has finished synchronizing until you have checked all relevant log files including base and child channels.

Create an activation key associated with the Red Hat Enterprise Linux channel.

Creating a Bootstrap Repository

The following procedure demonstrate creating a bootstrap repository for RHEL:

1. On the server command line as root, create a bootstrap repo for RHEL with the following command:

```
mgr-create-bootstrap-repo RHEL_activation_channel_key
```

   If you use a dedicated channel per RHEL version, specify it with the `--with-custom-channel` option.

2. Rename `bootstrap.sh` to `resversion-boostrap.sh`:

```
{prompt.root}cp bootstrap.sh res7-bootstrap.sh
```

Register a Salt Minion via Bootstrap

The following procedure will guide you through registering a Salt minion using the bootstrap script.

Procedure: Registration Using the Bootstrap Script

1. For your new minion download the bootstrap script from the Uyuni server:

```
wget --no-check-certificate https://`server`/pub/bootstrap/res7-bootstrap.sh
```

2. Add the appropriate res-gpg-pubkey--.key to the `ORG_GPG_KEY` key parameter, comma delimited in your `res7-bootstrap.sh` script. These are located on your Uyuni server at:

```
http://`server`/pub/
```

3. Make the `res7-bootstrap.sh` script executable and run it. This will install necessary Salt packages from the bootstrap repository and start the Salt minion service:

```
{prompt.root}chmod +x res7-bootstrap.sh{prompt.root}./res7-boostrap.sh
```

4. From the Uyuni Web UI select **Salt › Keys** and accept the new minion's key.

Troubleshooting Bootstrap

If bootstrapping a minion fails it is usually caused by missing packages. These missing packages are contained on the RHEL installation media. The RHEL installation media should be loop mounted and added as a child channel to the Red Hat Enterprise Linux channel. See the warning in [bp.expanded-support.resclients] on access to RHEL Media.

Manual Salt Minion Registration

The following procedure will guide you through the registration of a Salt minion manually.

1. Add the bootstrap repository:

```
yum-config-manager --add-repo https://`server`/pub/repositories/res/7/bootstrap
```

2. Install the salt-minion package:

```
{prompt.root}yum install salt-minion
```

3. Edit the Salt minion configuration file to point to the Uyuni server:

```
{prompt.root}mkdir /etc/salt/minion.d{prompt.root}echo "master:`server_fqdn`" >
/etc/salt/minion.d/susemanager.conf
```

4. Start the minion service:

```
{prompt.root}systemctl start salt-minion
```

5. From the Uyuni Web UI select the **Salt › Keys** and accept the new minion's key.

Preparing Channels and Repositories for CentOS Traditional Clients

This following section provides an example procedure for configuring CentOS channels and repositories and finally registering a CentOS client with Uyuni.

These steps will be identical for Scientific Linux and Fedora.

Procedure: Preparing Channels and Repositories

1. As root install spacewalk-utils on your Uyuni server:

```
zypper in spacewalk-utils
```

Supported Tools

The spacewalk-utils package contains a collection of upstream command line tools which provide assistance with spacewalk administrative operations. You will be using the `spacewalk-common-channels` tool. Keep in mind SUSE only provides support for `spacewalk-clone-by-date` and `spacewalk-manage-channel-lifecycle` tools.

2. Run the `spacewalk-common-channels` script to add the CentOS7 base, updates, and Spacewalk client channels.

```
{prompt.root}spacewalk-common-channels -u admin -p`secret`-a x86_64
'centos7'{prompt.root}spacewalk-common-channels -u admin -p`secret`-a x86_64 'centos7-
updates'{prompt.root}spacewalk-common-channels -u admin -p`secret`-a x86_64
'spacewalk26-client-centos7'
```

Required Channel References

The `/etc/rhn/spacewalk-common-channels.ini` must contain the channel references to be added. If a channel is not listed, check the latest version here for updates: https://github.com/spacewalkproject/spacewalk/tree/master/utils

3. From the Web UI select **Main Menu › Software › Manage Software Channels › Overview**. Select the base channel you want to synchronize, in this case `CentOS7 (x86_64)`. Select **Repositories › Sync**. Check the channels you want to synchronize and then click the **[Sync Now]** button or, optionally, schedule a regular synchronization time.

4. Copy all relevant GPG keys to `/srv/www/htdocs/pub`. Depending on what distribution you are interested in managing these could include an EPEL key, SUSE keys, Red Hat keys, and CentOS keys. After copying these you can reference them in a comma-delimited list within your bootstrap script (see Procedure: Preparing the Bootstrap Script).
    - CentOS7 key files: http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-7
    - EPEL key file: http://mirrors.kernel.org/fedora-epel/RPM-GPG-KEY-EPEL-7

- Spacewalk key: http://spacewalk.redhat.com/yum/RPM-GPG-KEY-spacewalk-2015
- Red Hat keys: http://www.redhat.com/contact/security-response-team/gpg-keys.html

5. Install and setup a CentOS 7 client with the default installation packages.
6. Ensure the client machine can resolve itself and your Uyuni server via DNS. Validate that there is an entry in `/etc/hosts` for the real IP address of the client.
7. Create an activation key (`centos7`) on the Uyuni server that points to the correct parent/child channels, including the CentOS base repo, updates, and Spacewalk client.

Now prepare the bootstrap script.

Procedure: Preparing the Bootstrap Script

1. Create/edit your bootstrap script to correctly reflect the following:

```
# can be edited, but probably correct (unless created during initial install):

# NOTE: ACTIVATION_KEYS *must* be used to bootstrap a client machine.

ACTIVATION_KEYS=1-centos7

ORG_GPG_KEY=res.key,RPM-GPG-KEY-CentOS-7,suse-307E3D54.key,suse-9C800ACA.key,RPM-GPG-
KEY-spacewalk-2015

FULLY_UPDATE_THIS_BOX=0

yum clean all
# Install the prerequisites
yum -y install yum-rhn-plugin rhn-setup
```

2. Add the following lines to the bottom of your script, (just before `echo "-bootstrap complete -"`):

```
# This section is for commands to be executed after registration
mv /etc/yum.repos.d/Cent* /root/
yum clean all
chkconfig rhnsd on
chkconfig osad on
service rhnsd restart
service osad restart
```

3. Continue by following normal bootstrap procedures to bootstrap the new client.

Registering CentOS Salt Minions with Uyuni

The following procedure will guide you through registering a CentOS Minion.

Support for CentOS Patches

CentOS uses patches originating from CentOS is not officially supported by SUSE . See the matrix of Uyuni clients on the main page of the Uyuni wiki, linked from the *Quick Links* section: https://wiki.microfocus.com/index.php?title=SUSE_Manager

Procedure: Register a CentOS 7 Minion

1. Add the Open Build Service repo for Salt:

```
{prompt.root}yum-config-manager --add-repo
http://download.opensuse.org/repositories/systemsmanagement:/saltstack:/products/RHEL_7/
```

2. Import the repo key:

```
{prompt.root}rpm --import
http://download.opensuse.org/repositories/systemsmanagement:/saltstack:/products/RHEL_7/
repodata/repomd.xml.key
```

3. Check if there is a different repository that contains Salt. If there is more than one repository listed disable the repository that contains Salt apart from the OBS one.

```
{prompt.root}yum list --showduplicates salt
```

4. Install the Salt minion:

```
{prompt.root}yum install salt salt-minion
```

5. Change the Salt configuration to point to the Uyuni server:

```
{prompt.root}mkdir -p /etc/salt/minion.d{prompt.root}echo "master:`server_fqdn`" >
/etc/salt/minion.d/susemanager.conf
```

6. Restart the minion

```
{prompt.root}systemctl restart salt-minion
```

7. Proceed to **Main Menu › Salt › Keys** from the Web UI and accept the minion's key.

Ubuntu Clients

Support for Ubuntu Clients was added in SUSE Manager 3.2. Currently, Salt minions running Ubuntu 16.04 LTS and 18.04 LTS are supported.

Ubuntu clients must be Salt minions. Traditional clients are not supported.

Bootstrapping is supported for starting Ubuntu clients and performing initial state runs such as setting repositories and performing profile updates. However, the root user on Ubuntu is disabled by default, so in order to use bootstrapping, you will require an existing user with `sudo` privileges for Python.

Other supported features:

- Synchronizing `.deb` channels
- Assigning `.deb` channels to minions
- GPG signing `.deb` repositories
- Information displayed in System details pages
- Package install, update, and remove
- Package install using `Package States`
- Configuration and state channels

Some actions are not yet supported:

- Patch and errata support
- Bare metal installations, PXE booting, and virtual host provisioning
- Live patching
- CVE Audit
- If you use are using a repository from storage media (`server.susemanager.fromdir = ...` option in rhn.conf), Ubuntu Client Tools will not work.

Prepare to Register Ubuntu Clients

Some preparation is required on before you can register Ubuntu clients to Uyuni Server.

Before you begin, ensure you have the Ubuntu product enabled, and have synchronized the Ubuntu channels:

For Ubuntu 18.04:

- Product: Ubuntu Client 18.04
- Mandatory channels: `ubuntu-18.04-pool-amd64`

For Ubuntu 16.04:

- Product: Ubuntu Client 16.04
- Mandatory channels: `ubuntu-16.04-pool-amd64`

The mandatory channels do not contain Ubuntu upstream packages. The repositories and channels for synchronizing upstream content must be configured manually.

Procedure: Preparing to Register Ubuntu Clients

1. Ensure that you have the appropriate software channels available on your system. In the Uyuni Web UI, navigate to **Software › Channel List › All**. You should see a base channel and a child channel for your architecture, for example:

```
ubuntu-18.04-pool for amd64
|
+- Ubuntu-18.04-SUSE-Manager-Tools for amd64
```

2. Create custom repositories to mirror the Ubuntu packages. For more information on creating custom repositories, see modules/administration/pages/channel-management.pdf. For example:

   For `main`:

   - Repository Label: ubuntu-bionic-main
   - Repository URL: http://ubuntumirror.example.com/ubuntu/dists/bionic/main/binary-amd64/
   - Repository Type: deb

   For `main-updates`:

   - Repository Label: ubuntu-bionic-main-updates
   - Repository URL: http://ubuntumirror.example.com/ubuntu/dists/bionic-updates/main/binary-amd64/
   - Repository Type: deb

3. Create custom channels under the `pool` channel, mirroring the vendor channels. For more information on creating custom channels, see modules/administration/pages/channel-management.pdf. Ensure the custom channels you create have `AMD64 Debian` architecture.

   For example:

```
ubuntu-18.04-pool for amd64 (vendor channel)
|
+- Ubuntu-18.04-SUSE-Manager-Tools for amd64 (vendor channel)
|
+- ubuntu-18.04-amd64-main (custom channel)
|
+- ubuntu-18.04-amd64-main-updates (custom channel)
```

4. Associate the custom channels with the appropriate custom repositories.
5. Synchronize the new custom channels. You can check the progress of your synchronization from the command line with this command:

```
tail -f /var/log/rhn/reposync.log /var/log/rhn/reposync/*
```

1. To use bootstrap with Ubuntu, you will need to create a bootstrap repository. You can do this from the command line with `mgr-create-bootstrap-repo`:

   ```
   mgr-create-bootstrap-repo --with-custom-channels
   ```

The root user on Ubuntu is disabled by default. You can enable it by editing the `sudoers` file.

Procedure: Granting Root User Access

1. On the client, edit the `sudoers` file:

   ```
   sudo visudo
   ```

   Grant `sudo` access to the user by adding this line to the `sudoers` file. Replace `<user>` with the name of the user that will be used to bootstrap the client in the Web UI:

   ```
   <user>   ALL=NOPASSWD: /usr/bin/python, /usr/bin/python2, /usr/bin/python3
   ```

Activation Keys

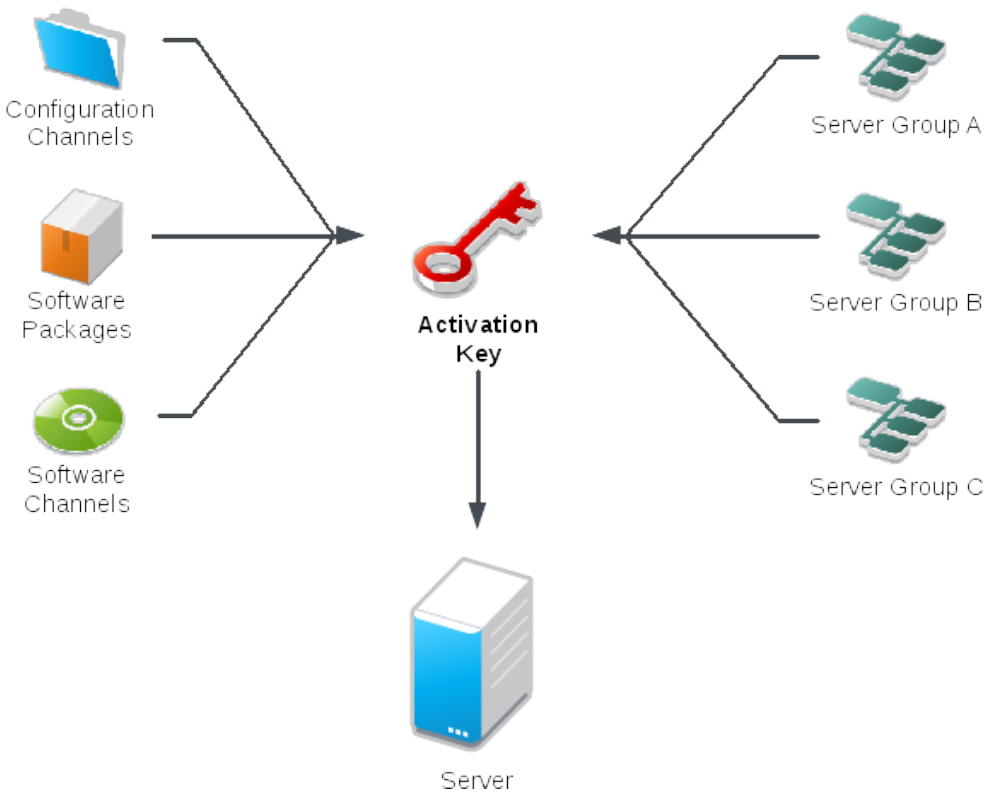
Activation keys are used with traditional and Salt clients to ensure that your clients have the correct software entitlements, are connecting to the appropriate channels, and are subscribed to the relevant groups. Each activation key is bound to an organization, which you can set when you create the key.

In Uyuni, an activation key is a group of configuration settings with a label. You can apply all configuration settings associated with an activation key by adding its label as a parameter to a bootstrap script. Under normal operating conditions best practices suggest using an activation key label in combination with a bootstrap script.

An activation key can specify:

- Channel Assignment
- System Types (Traditionally called Add-on Entitlements)
- Contact Method
- Configuration Files
- Packages to be Installed
- System Group Assignment

Activation keys are a collection of configuration settings which have been given a label name and then added to a bootstrap script. When the bootstrap script is executed all configuration settings associated with the label are applied to the system the script is run on.

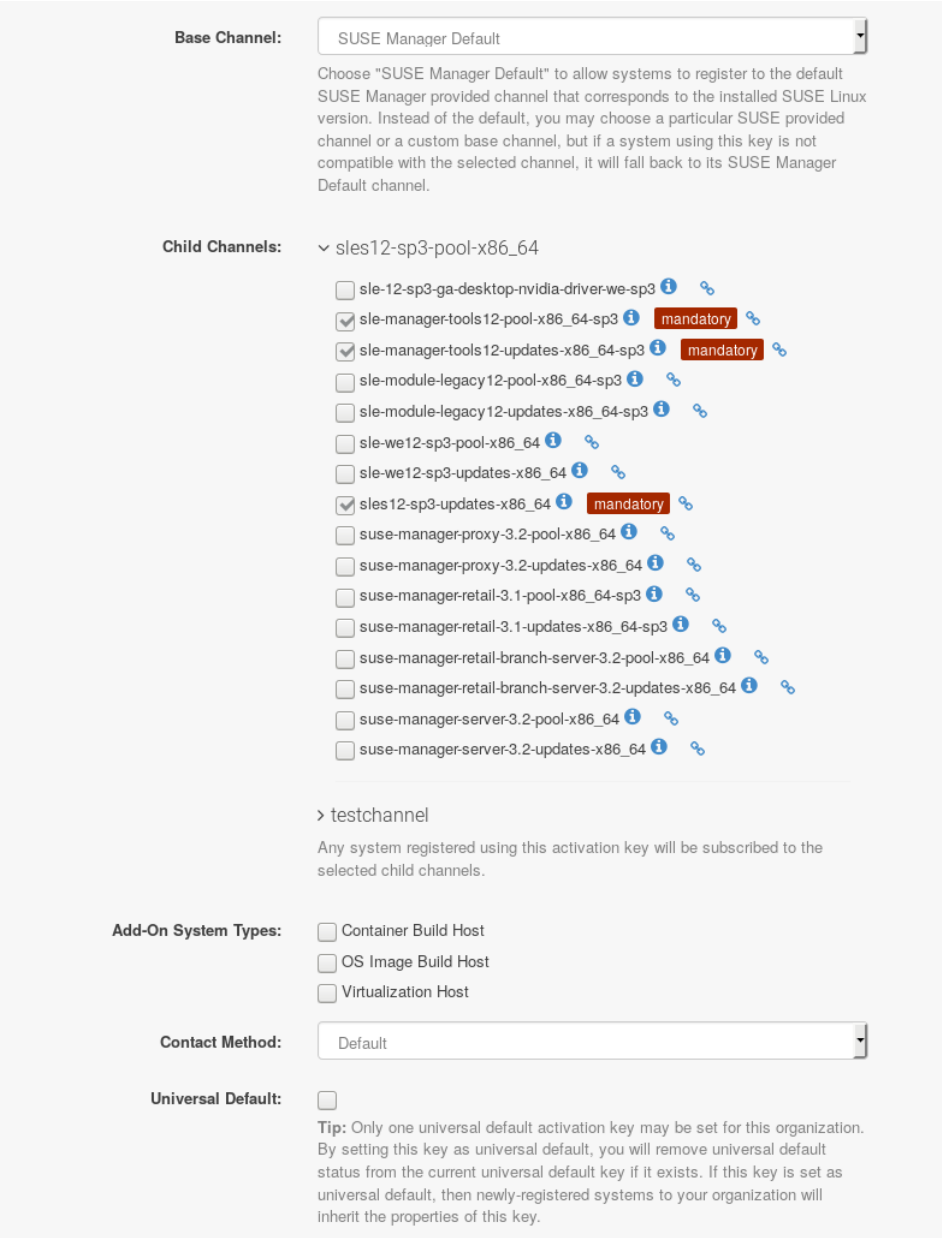


Procedure: Creating Activation Keys

1. Log in to the Uyuni Web UI as an administrator, and navigate to **Systems › Activation Keys**.
2. Click the **[Create Key]** button.
3. On the `Activation Key Details` page, in the `Description` field, enter a name for the activation key.
4. In the `Key` field, enter the distribution and service pack associated with the key. For example, `SLES12-SP4` for SUSE Linux Enterprise Server 12 SP4.

Do not use commas in the `Key` field for any SUSE products. However, you **must** use commas for Red Hat Products. For more information, see xref:FILENAME.adoc#ref.webui.systems.activ-keys[].

5. In the `Base Channels` drop-down box, select the appropriate base software channel, and allow the relevant child channels to populate.
6. Select the child channels you need (for example, the mandatory SUSE Manager tools and updates channels).



7. We recommend you leave the `Contact Method` set to `Default`.
8. We recommend you leave the `Universal Default` setting unchecked.
9. Click **[Update Activation Key]** to create the activation key.
10. Check the `Configuration File Deployment` check box to enable configuration management for this key, and click **[Update Activation Key]** to save this change.

Note that the `Configuration File Deployment` check box does not appear until after you have created the activation key. Ensure you go back and check the box if you need to enable configuration management.

Activation Key Best Practices

Default Parent Channel

Avoid using the `SUSE Manager Default` parent channel. This setting forces Uyuni to choose a parent channel that best corresponds to the installed operating system, which can sometimes lead to unexpected behavior. Instead, we recommend you create activation keys specific to each distribution and architecture.

Bootstrapping with Activation Keys

If you are using bootstrap scripts, consider creating an activation key for each script. This will help you align channel assignments, package installation, system group memberships, and configuration channel assignments. You will also need less manual interaction with your system after registration.

Bandwidth Requirements

Using activation keys might result in automatic downloading of software at registration time, which might not be desirable in environments where bandwidth is constrained.

These options create bandwidth usage:

- Assigning a SUSE Product Pool channel will result in the automatic installation of the corresponding product descriptor package.
- Any package in the `Packages` section will be installed.
- Any Salt state from the `Configuration` section might trigger downloads depending on its contents.

Key Label Naming

If you do not enter a human-readable name for your activation keys, the system will automatically generate a number string, which can make it difficult to manage your keys.

Consider a naming scheme for your activation keys to help you keep track of them. Creating names which are associated with your organization's infrastructure will make it easier for you when performing more complex operations.

When creating key labels, consider these tips:

- OS naming (mandatory): Keys should always refer to the OS they provide settings for
- Architecture naming (recommended): Unless your company is running on one architecture only, for example x86_64, then providing labels with an architecture type is a good idea.
- Server type naming: What is, or what will this server be used for?
- Location naming: Where is the server located? Room, building, or department?
- Date naming: Maintenance windows, quarter, etc.
- Custom naming: What naming scheme suits your organizations needs?

Example activation key label names:

```
sles12-sp2-web_server-room_129-x86_64
```

```
sles12-sp2-test_packages-blg_502-room_21-ppc64le
```

Do not use commas in the `Key` field for any SUSE products. However, you **must** use commas for Red Hat Products. For more information, see xref:FILENAME.adoc#ref.webui.systems.activ-keys[].

Included Channels

When creating activation keys you also need to keep in mind which software channels will be associated with it.

Default Base Channel

Keys should have a specific base channel assigned to it, for example `SLES12-SP2-Pool-x86_64`. If this is not the case Uyuni cannot use specific stages. Using the default base channel is not recommended and may cause problems.

- Channels to be included:
  - suse-manager-tools
- Typical packages to be included:
  - mgr-osad (pushing tasks)
    - Installs python-jabberpy and pyxml as dependencies
  - mgr-cfg-actions (Remote Command, Configuration Managment)
    - Installs mgr-cfg and mgr-cfg-client as dependencies

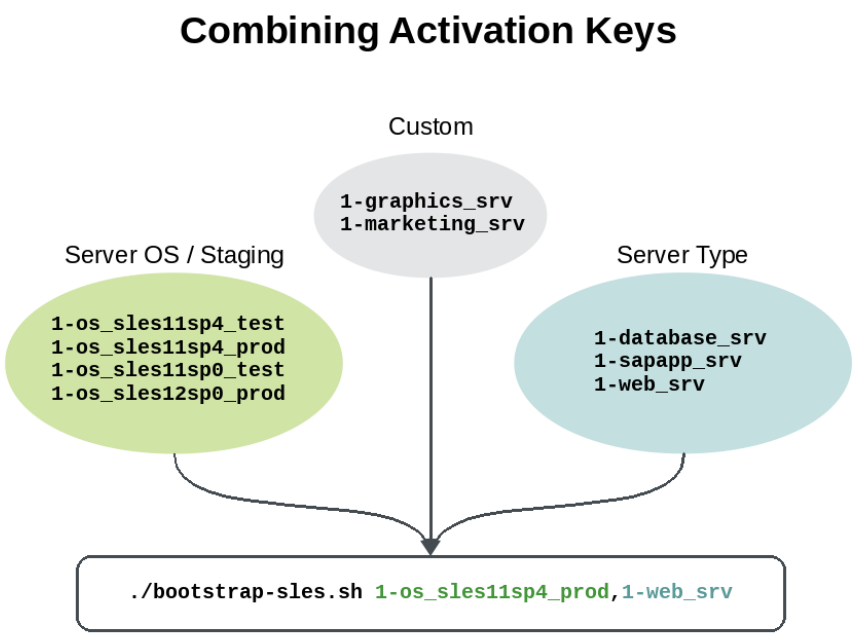
The `suse-manager-tools` channel is mandatory.

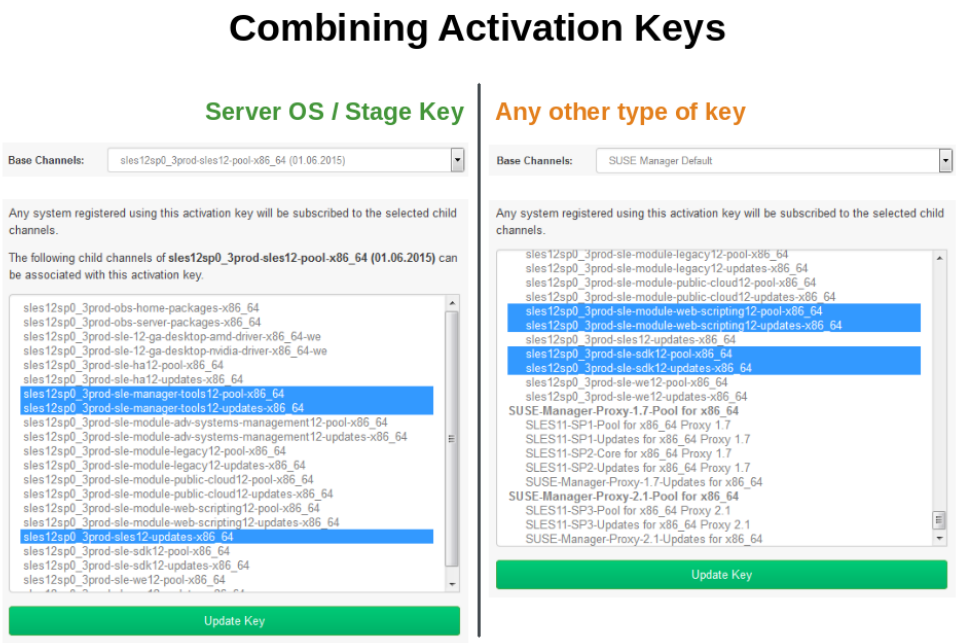
Typical packages to be included:

- osad (pushing tasks): Installs `python-jabberpy` and `pyxml` as dependencies
- `rhncfg-actions` (Remote Command, Configuration Managment): Installs `rhncfg` and `rhncfg-client` as dependencies

Combining Activation Keys

You can combine activation keys when executing the bootstrap script on your clients. Combining keys allows for more control on what is installed on your systems and reduces duplication of keys for large or complex environments.





Using Activation Keys with Traditional Clients

Create the initial bootstrap script template from the command line on the Uyuni server with:

```
# mgr-bootstrap
```

This command will generate the bootstrap script and place them in `/srv/www/htdocs/pub/bootstrap`.

Alternatively, you can use the Uyuni Web UI to create your bootstrap script template, from **Overview** › **Tasks**.

For more information, see registering.clients.bootstrap.

Software Channels

Channels are a method of grouping software packages. In Uyuni channels are divided into base channels, and child channels. Organizing channels in this way ensures that only compatible packages are installed on each system.

A base channel consists of packages built for a specific operating system type, version, and architecture. For example, all of the packages in SUSE Linux Enterprise Server 12 for the `x86_64` architecture make up a base channel. The list of packages in SUSE Linux Enterprise Server 12 for the `s390x` architecture make up a different base channel. A system must be subscribed to only one base channel assigned automatically during registration based on the SUSE Linux Enterprise release and system architecture. For paid channels provided by a vendor, you must have an associated subscription.

A child channel is associated with a specific base channel and provides only packages that are compatible with that base channel. A system can be subscribed to multiple child channels of its base channel. When a system has been assigned to a base channel, it is only possible for that system to install the related child channels. For example, if a system has been assigned to the SUSE Linux Enterprise Server 12 `x86_64` base channel, they will only be able to install or update packages compatible with SUSE Linux Enterprise Server 12 `x86_64`.

In the Uyuni Web UI you can browse your available channels by navigating to **Software › Channels**. You can modify or create new channels by navigating to **Software › Manage Software Channels**.

Custom Channels

If you require packages that are not provided by the standard Uyuni base channels, you can create custom channels. Uyuni Administrators and Channel Administrators have channel management authority, which gives them the ability to create and manage their own custom channels.

For more on creating custom channels, see the Administration Guide.

Creating the Uyuni Tools Repository

In this section you will create a tools repository on the Uyuni Server for providing client tools. The client tools repository contains packages for installing Salt on minions as well as required packages for registering traditional clients during the bootstrapping procedure. These packages will be installed from the newly generated repository during the registration process. In the following procedure you will create the SUSE Linux Enterprise tools repository.

Creating a Tools Repository when an SCC Channel has not been Synced

Before following the procedure to create the tools repository make sure the SUSE vendor channel you will be using with your client has been completely synced. You can check this by running `tail -f /var/log/rhn/reposync/<CHANNEL_NAME>.log` as *root*. In the following example replace `version` with the actual version string:

```
# tail -f /var/log/rhn/reposync/sles`version`-pool-x86_64.log
```

Once completed you should see the following output in your terminal:

```
2017/12/12 15:20:32 +02:00 Importing packages started.
2017/12/12 15:22:02 +02:00 1.07 %
...
2017/12/12 15:34:25 +02:00 86.01 %
2017/12/12 15:35:49 +02:00 Importing packages finished.
2017/12/12 15:35:49 +02:00 Linking packages to channel.
...
2017/12/12 15:35:59 +02:00 Sync completed.
```

Procedure: Generating the Tools Repository for SUSE Linux Enterprise

1. Open a terminal on the server as root and enter the following command to list available bootstrap repositories:

```
mgr-create-bootstrap-repo -l SLE-`version`-x86_64
```

2. Then invoke the same command using the listed repository as the product label to actually create the bootstrap repository:

```
mgr-create-bootstrap-repo -c SLE-`version`-x86_64
```

3. Uyuni will create and add the client tools to the newly created `repositories` directory located at `/srv/www/htdocs/pub/repositories/`.

This repository is suitable for both Server and Desktop of SUSE Linux Enterprise.

Support for SUSE Linux Enterprise 15 Products

If you have mirrored more than one SUSE Linux Enterprise 15 Product (for example, SLES and SLES for SAP Application), you can specify the one you are actually interested in. First check what is avaiable:

```
mgr-create-bootstrap-repo -c SLE-15-x86_64 --with-custom-channel
Multiple options for parent channel found. Please use option
--with-parent-channel <label> and choose one of:
- sle-product-sles15-pool-x86_64
- sle-product-sles_sap15-pool-x86_64
- sle-product-sled15-pool-x86_64
```

Then specify it with `--with-parent-channel`:

```
mgr-create-bootstrap-repo -c SLE-15-x86_64 --with-parent-channel sle-
product-sled15-pool-x86_64
```

Contact Methods

Selecting a Contact Method

Uyuni provides several methods for communication between client and server. All commands your Uyuni server sends its clients to do will be routed through one of them. Which one you select will depend on your network infrastructure. The following sections provide a starting point for selecting a method which best suits your network environment.

Contact Methods and Salt

This chapter is only relevant for traditional clients as Salt clients (minions) utilize a Salt specific contact method. For general information about Salt clients, see [salt.gs.guide.intro].

Default (the Uyuni Daemon rhnsd)

The Uyuni daemon (`rhnsd`) runs on client systems and periodically connects with Uyuni to check for new updates and notifications. The daemon, which runs in the background, is started by `rhnsd.service` . By default, it will check every 4 hours for new actions, therefore it may take some time for your clients to begin updating after actions have been scheduled for them.

To check for updates, `rhnsd` runs the external `mgr_check` program located in `/usr/sbin/` . This is a small application that establishes the network connection to Uyuni. The SUSE Manager daemon does not listen on any network ports or talk to the network directly. All network activity is done via the `mgr_check` utility.

Auto accepting (EULAs)

When new packages or updates are installed on the client using Uyuni, any end user licence agreements (EULAs) are automatically accepted. To review a package EULA, open the package detail page in the Web UI.

This figure provides an overview of the default `rhnsd` process path. All items left of the `Python XMLRPC server` block represent processes running on a Uyuni client.

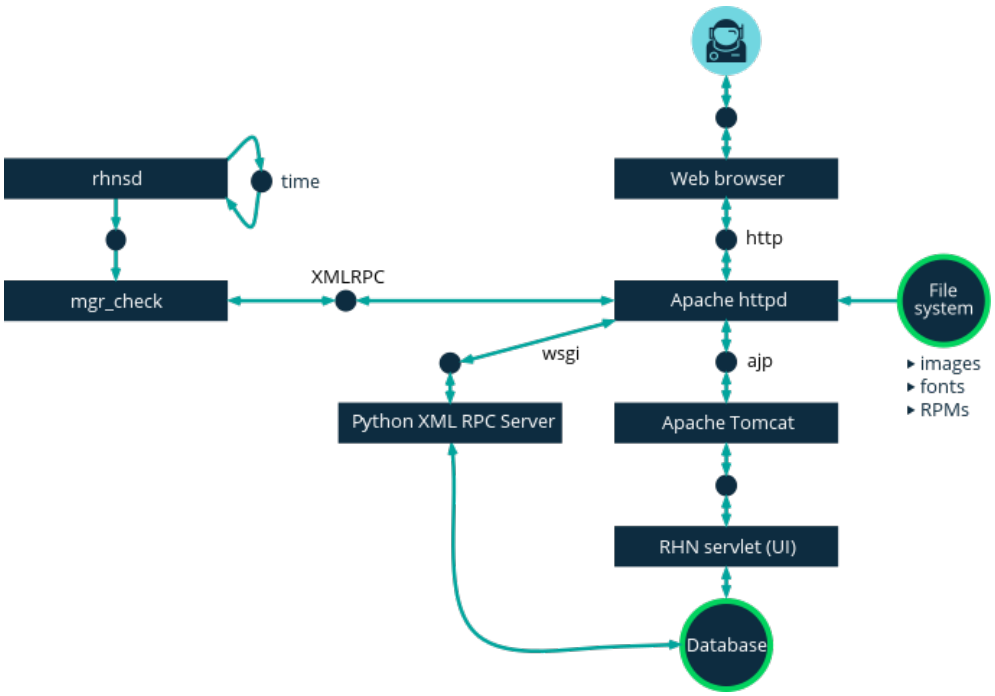


Figure 1. rhnsd Contact Method

Configuring Uyuni rhnsd Daemon

The Uyuni daemon can be configured by editing the file on the client:

```
/etc/sysconfig/rhn/rhnsd
```

This is the configuration file the rhnsd initialization script uses. An important parameter for the daemon is its check-in frequency. The default interval time is four hours (240 minutes). If you modify the configuration file, you must as root restart the daemon with `systemctl rhnsd restart`.

Minimum Allowed Check-in Parameter

The minimum allowed time interval is one hour (60 minutes). If you set the interval below one hour, it will change back to the default of 4 hours (240 minutes).

Viewing rhnsd Daemon Status

You can view the status of rhnsd by typing the command `systemctl status rhnsd` as root .

Push via SSH

Push via SSH is intended to be used in environments where your clients cannot reach the Uyuni server directly to regularly check in and, for example, fetch package updates.

In detail, this feature enables a Uyuni located within an internal network to manage clients located on a “Demilitarized Zone” (DMZ) outside of the firewall protected network. Due to security reasons, no

system on a DMZ is authorized to open a connection to the internal network and therefore your Uyuni server. The solution is to configure Push via SSH which utilizes an encrypted tunnel from your Uyuni server on the internal network to the clients located on the DMZ. After all actions/events are executed, the tunnel is closed. The server will contact the clients in regular intervals (using SSH) to check in and perform all actions and events.

Push via SSH Unsupported Actions

Certain actions are currently not supported on scheduled clients which are managed via Push via SSH. This includes re-installation of systems using the provisioning module.

The following figure provides an overview of the Push via SSH process path. All items left of the `Taskomatic` block represent processes running on a Uyuni client.

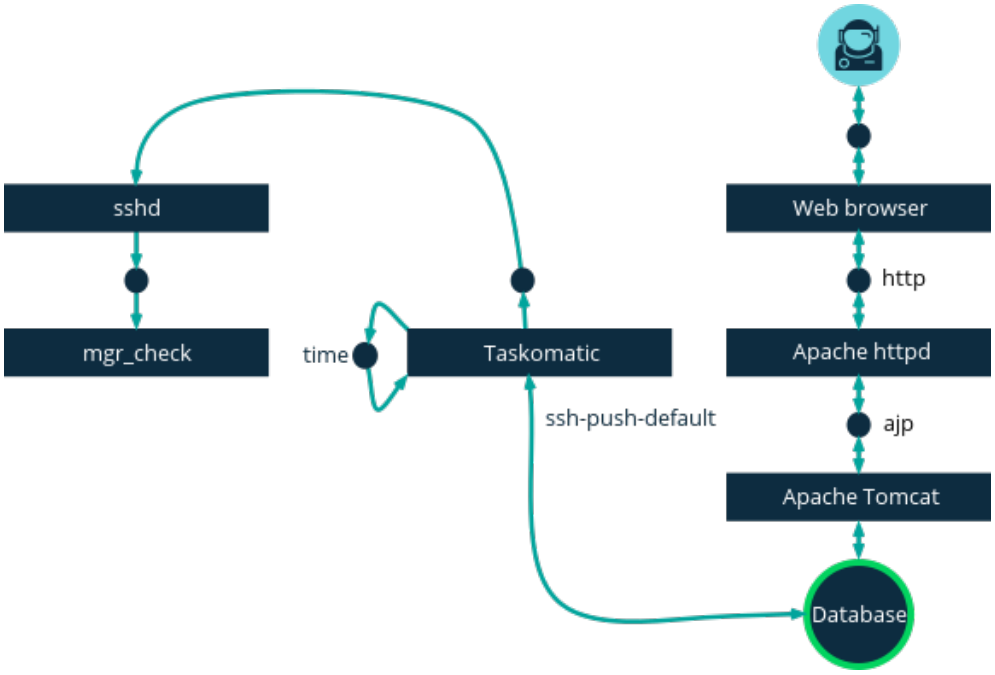


Figure 2. Push via SSH Contact Method

Configuring the Server for Push via SSH

For tunneling connections via SSH, two available port numbers are required, one for tunneling HTTP and the second for tunneling via HTTPS (HTTP is only necessary during the registration process). The port numbers used by default are `1232` and `1233`. To overwrite these, add two custom port numbers greater than 1024 to `/etc/rhn/rhn.conf` like this:

```
ssh_push_port_http = high port 1
ssh_push_port_https = high port 2
```

If you would like your clients to be contacted via their hostnames instead of an IP address, set the following option:

```
ssh_push_use_hostname = true
```

It is also possible to adjust the number of threads to use for opening client connections in parallel. By default two parallel threads are used. Set `taskomatic.ssh_push_workers` in `/etc/rhn/rhn.conf` like this:

```
taskomatic.ssh_push_workers = number
```

Using sudo with Push via SSH

For security reasons you may desire to use sudo and SSH into a system as a user other than root . The following procedure will guide you through configuring sudo for use with Push via SSH.

sudo Requirements

The packages `spacewalk-taskomatic >= 2.1.165.19` and `spacewalk-certs-tools ⇒ 2.1.6.7` are required for using sudo with Push via SSH.

Procedure: Configuring sudo

1. Set the following parameter on the server located in `/etc/rhn/rhn.conf` .

   ```
   ssh_push_sudo_user =`user`
   ```

   The server will use sudo to ssh as the configured `user`.

2. You must create the user specified in Procedure: Configuring sudo on each of your clients and the following parameters should be commented out within each client's `/etc/sudoers` file:

   ```
   #Defaults targetpw   # ask for the password of the target user i.e. root
   #ALL    ALL=(ALL) ALL   # WARNING! Only use this together with 'Defaults targetpw'!
   ```

3. Add the following lines beneath the `\## User privilege specification` section of each client's `/etc/sudoers` file:

   ```
   <user> ALL=(ALL) NOPASSWD:/usr/sbin/mgr_check
   <user> ALL=(ALL) NOPASSWD:/home/<user>/enable.sh
   <user> ALL=(ALL) NOPASSWD:/home/<user>/bootstrap.sh
   ```

4. On each client add the following two lines to the `/home/user/.bashrc` file:

```
PATH=$PATH:/usr/sbin
export PATH
```

Client Registration

As your clients cannot reach the server, you will need to register your clients from the server. A tool for performing registration of clients from the server is included with Uyuni and is called `mgr-ssh-push-init`. This tool expects a client's hostname or IP address and the path to a valid bootstrap script located in the server's filesystem for registration as parameters.

Specifying Ports for Tunneling before Registering Clients

The ports for tunneling need to be specified before the first client is registered. Clients already registered before changing the port numbers must be registered again, otherwise the server will not be able to contact them anymore.

`mgr-ssh-push-init` *Disables rhnsd*

The `mgr-ssh-push-init` command disables the `rhnsd` daemon which normally checks for updates every 4 hours. Because your clients cannot reach the server without using the Push via SSH contact method, the `rhnsd` daemon is disabled.

For registration of systems which should be managed via the Push via SSH tunnel contact method, it is required to use an activation key that is configured to use this method. Normal `Push via SSH` is unable to reach the server. For managing activation keys, see [bp.key.managment].

Run the following command as root on the server to register a client:

```
# mgr-ssh-push-init --client client --register \
/srv/www/htdocs/pub/bootstrap/bootstrap_script --tunnel
```

To enable a client to be managed using Push via SSH (without tunneling), the same script may be used. Registration is optional since it can also be done from within the client in this case. `mgr-ssh-push-init` will also automatically generate the necessary SSH key pair if it does not yet exist on the server:

```
# mgr-ssh-push-init --client`client`--register bootstrap_script
```

When using the Push via SSH tunnel contact method, the client is configured to connect to Uyuni using the high ports mentioned above. Tools like `rhn_check` and `zypper` will need an active SSH session with the proper port forwarding options in order to access the Uyuni API. To verify the Push via SSH tunnel connection manually, run the following command on the Uyuni server:

```
# ssh -i /root/.ssh/id_susemanager -R high port: susemanager :443`client`zypper ref
```

API Support for Push via SSH

The contact method to be used for managing a server can also be modified via the API. The following example code (python) shows how to set a system's contact method to `ssh-push`. Valid values are:

- `default` (pull)
- `ssh-push`
- `ssh-push-tunnel`

```
client = xmlrpclib.Server(SUMA_HOST + "/rpc/api", verbose=0)
key = client.auth.login(SUMA_LOGIN, SUMA_PASSWORD)
client.system.setDetails(key, 1000012345, {'contact_method' : 'ssh-push'})
```

Migration and Management via Push via SSH

When a system should be migrated and managed using Push via SSH, it requires setup using the `mgr-ssh-push-init` script before the server can connect via SSH. This separate command requires human interaction to install the server's SSH key onto the managed client (root password). The following procedure illustrates how to migrate an already registered system:

Procedure: Migrating Registered Systems

1. Setup the client using the `mgr-ssh-push-init` script (without `--register`).
2. Change the client's contact method to `ssh-push` or `ssh-push-tunnel` respectively (via API or Web UI).

Existing activation keys can also be edited via API to use the Push via SSH contact method for clients registered with these keys:

```
client.activationkey.setDetails(key, '1-mykey', {'contact_method' : 'ssh-push'})
```

Proxy Support with Push via SSH

It is possible to use Push via SSH to manage systems that are connected to the Uyuni server via a proxy. To register a system, run `mgr-ssh-push-init` on the proxy system for each client you wish to register. Update your proxy with the latest packages to ensure the registration tool is available. It is necessary to copy the ssh key to your proxy. This can be achieved by executing the following command from the server:

```
{prompt.root}mgr-ssh-push-init --client`proxy`
```

Push via Salt SSH

Push via Salt SSH is intended to be used in environments where your Salt clients cannot reach the Uyuni server directly to regularly checking in and, for example, fetch package updates.

Push via SSH

This feature is not related to Push via SSH for the traditional clients. For Push via SSH, see Push via SSH.

Overview

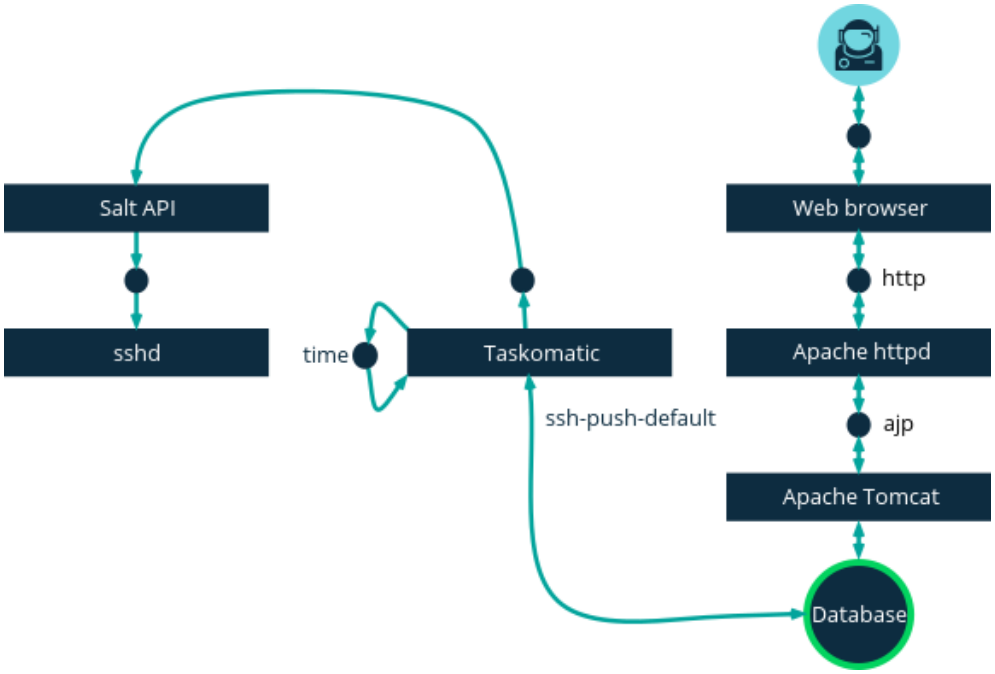


Figure 3. Push via Salt SSH Contact Method

Salt provides “Salt SSH” (`salt-ssh`), a feature to manage clients from a server. It works without installing Salt related software on clients. Using Salt SSH there is no need to have minions connected to the Salt master. Using this as a Uyuni connect method, this feature provides similar functionality for Salt clients as the traditional Push via SSH feature for traditional clients.

This feature allows:

- Managing Salt entitled systems with the Push via SSH contact method using Salt SSH.
- Bootstrapping such systems.

Requirements

- SSH daemon must be running on the remote system and reachable by the `salt-api` daemon (typically running on the Uyuni server).
- Python must be available on the remote system (Python must be supported by the installed Salt).

Currently: python 2.6.

Unsupported Systems

Red Hat Enterprise Linux and CentOS versions ⇐ 5 are not supported because they do not have Python 2.6 by default.

Bootstrapping

To bootstrap a Salt SSH system, proceed as follows:

1. Open the **Bootstrap Minions ›] dialog in the Web UI (menu:Systems[Bootstrapping**).
2. Fill out the required fields. Select an **Activation Key ›] with the menu:Push via SSH[** contact method configured. For more information about activation keys, see [ref.webui.systems.activ-keys].
3. Check the **Manage system completely via SSH** option.
4. Confirm with clicking the **Bootstrap** button.

Now the system will be bootstrapped and registered in Uyuni. If done successfully, it will appear in the **Systems** list.

Configuration

There are two kinds of parameters for Push via Salt SSH:

- Bootstrap-time parameters - configured in the **Bootstrapping** page:
  - Host
  - Activation key
  - Password - used only for bootstrapping, not saved anywhere; all future SSH sessions are authorized via a key/certificate pair
- Persistent parameters - configured Uyuni-wide:
  - sudo user - same as in Using sudo with Push via SSH.

Action Execution

The Push via Salt SSH feature uses a taskomatic job to execute scheduled actions using `salt-ssh`. The taskomatic job periodically checks for scheduled actions and executes them. While on traditional clients with SSH push configured only `rhn_check` is executed via SSH, the Salt SSH push job executes a complete `salt-ssh` call based on the scheduled action.

Known Limitation

- OpenSCAP auditing is not available on Salt SSH minions.
- Beacons do not work with Salt SSH.

- Installing a package on a system using `zypper` will not invoke the package refresh.
- Virtual Host functions (for example, a host to guests) will not work if the virtual host system is Salt SSH-based.

For More Information

For more information, see

- https://wiki.microfocus.com/index.php/SUSE_Manager/SaltSSHServerPush
- https://docs.saltstack.com/en/latest/topics/ssh/

OSAD

OSAD is an alternative contact method between Uyuni and its clients. By default, Uyuni uses `rhnsd`, which contacts the server every four hours to execute scheduled actions. OSAD allows registered client systems to execute scheduled actions immediately.

OSAD has several distinct components:

- The `osa-dispatcher` service runs on the server, and uses database checks to determine if clients need to be pinged, or if actions need to be executed.
- The `osad` service runs on the client. It responds to pings from `osa-dispatcher` and runs `mgr_check` to execute actions when directed to do so.
- The `jabberd` service is a daemon that uses the `XMPP` protocol for communication between the client and the server. The `jabberd` service also handles authentication.
- The `mgr_check` tool runs on the client to execute actions. It is triggered by communication from the `osa-dispatcher` service.

The `osa-dispatcher` periodically runs a query to check when clients last showed network activity. If it finds a client that has not shown activity recently, it will use `jabberd` to ping all `osad` instances running on all clients registered with your Uyuni server. The `osad` instances respond to the ping using `jabberd`, which is running in the background on the server. When the `osa-dispatcher` receives the response, it marks the client as online. If the `osa-dispatcher` fails to receive a response within a certain period of time, it marks the client as offline.

When you schedule actions on an OSAD-enabled system, the task will be carried out immediately. The `osa-dispatcher` periodically checks clients for actions that need to be executed. If an outstanding action is found, it uses `jabberd` to execute `mgr_check` on the client, which will then execute the action.

Enabling and Configuring OSAD

This section covers enabling the `osa-dispatcher` and `osad` services, and performing initial setup.

OSAD clients use the fully qualified domain name (FQDN) of the server to communicate with the `osa-dispatcher` service.

SSL is required for `osad` communication. If SSL certificates are not available, the daemon on your client systems will fail to connect. Make sure your firewall rules are set to allow the required ports. For more information, see [tab.install.ports.server].

Procedure: Enabling OSAD

1. On your Uyuni server, as the root user, start the `osa-dispatcher` service:

```
systemctl start osa-dispatcher
```

2. On each client machine, install the `mgr-osad` package from the `Tools` child channel. The `mgr-osad` package should be installed on clients only. If you install the `mgr-osad` package on your Uyuni Server, it will conflict with the `osa-dispatcher` package.
3. On the client systems, as the root user, start the `osad` service:

```
systemctl start osad
```

   Because `osad` and `osa-dispatcher` are run as services, you can use standard commands to manage them, including `stop`, `restart`, and `status`.

Configuration and Log Files

Each OSAD component is configured by local configuration files. We recommend you keep the default configuration parameters for all OSAD components.

Component	Location	Path to Configuration File
`osa-dispatcher`	Server	`/etc/rhn/rhn.conf` Section: `OSA configuration`
`osad`	Client	`/etc/sysconfig/rhn/osad.conf`
`osad` log file	Client	`/var/log/osad`

Component	Location	Path to Configuration File
`jabberd` log file	Both	`/var/log/messages`

Troubleshooting OSAD

If your OSAD clients cannot connect to the server, or if the `jabberd` service takes a lot of time responding to port 5552, it could be because you have exceeded the open file count.

Every client needs one always-open TCP connection to the server, which consumes a single file handler. If the number of file handlers currently open exceeds the maximum number of files that `jabberd` is allowed to use, `jabberd` will queue the requests, and refuse connections.

To resolve this issue, you can increase the file limits for `jabberd` by editing the `/etc/security/limits.conf` configuration file and adding these lines:

```
jabbersoftnofile5100
jabberhardnofile6000
```

Calculate the limits required for your environment by adding 100 to the number of clients for the soft limit, and 1000 to the current number of clients for the soft limit. In the example above, we have assumed 500 current clients, so the soft limit is 5100, and the hard limit is 6000.

You will also need to update the `max_fds` parameter in the `/etc/jabberd/c2s.xml` file with your chosen hard limit:

```
<max_fds>6000</max_fds>
```

Using the System Set Manager

System Set Manager is used to administrate groups of systems, rather than performing actions on one system at a time. It works for both Salt and Traditional systems.

For a complete list of the tasks that you can perform with the System Set Manager, see xref:ref.webui.systems.ssm.

Setting up System Set Manager

You need to select which systems you want to work with before you can use System Set Manager to perform operations.

Navigate to **Main Menu › Systems › System List › All** and check the boxes to the left of the systems you want to work with. This will automatically add your chosen systems to System Set Manager.

You can access System Set Manager in three different ways:

- Navigating to **Systems › System Set Manager**.
- Navigating to **Systems › System Groups** and clicking **[Use in SSM]** for the system group you want to work with.
- Navigating to **Systems › System Groups**, selecting the group you want to work with, and clicking **[Work with Group]**.

Using System Set Manager

The details you see in System Set Manager might differ slightly from the details available in other parts of the Uyuni Web UI. If you are looking at the details of a single system in the Web UI, then you will only be able to see the latest available versions of package updates. When you look at the same system in System Set Manager, all available versions will be shown. This is intended to make it easier for system administrators to manage package versions, and choose to upgrade to packages that might not be the latest version.

Troubleshooting Clients

Cloned Salt Clients

If you have used your hypervisor clone utility, and attempted to register the cloned Salt client, you might get this error:

```
We're sorry, but the system could not be found.
```

This is caused by the new, cloned, system having the same machine ID as an existing, registered, system. You can adjust this manually to correct the error and register the cloned system successfully.

For more information and instructions, see xref:FILENAME.adoc#bp.chapt.suma3.troubleshooting.registering.cloned.salt.systems[].

Mounting /tmp with noexec

Salt runs remote commands from `/tmp` of the client's filesystem. Therefore you must not mount `/tmp` with the `noexec` option.

SSL errors

On SLES 11 systems, clients can sometimes have SSL errors which make some operations unusable, including package management and bootstrapping. In this case, you will see an error like this:

```
Repository 'SLES11-SP4-SUSE-Manager-Tools x86_64' is invalid.
[|] Valid metadata not found at specified URL(s)
Please check if the URIs defined for this repository are pointing to a valid repository.
Skipping repository 'SLES11-SP4-SUSE-Manager-Tools x86_64' because of the above error.
Download (curl) error for 'www.example.com':
Error code: Unrecognized error
Error message: error:1409442E:SSL routines:SSL3_READ_BYTES:tlsv1 alert protocol version
```

This occurs because Apache requires TLS v1.2, but older versions of SLES do not support this version of the TLS protocol. To fix this error, you need to force Apache to accept a greater range of protocol versions. Open the `/etc/apache2/ssl-global.conf` configuration file, locate the `SSLProtocol` line, and update it to read:

```
SSLProtocol all -SSLv2 -SSLv3
```

This will need to be done manually on the Server, and with a Salt state on the Proxy. Restart the `apache` service on each system after making the changes.

GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections

then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
   Permission is granted to copy, distribute and/or modify this document
   under the terms of the GNU Free Documentation License, Version 1.2
   or any later version published by the Free Software Foundation;
   with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
   A copy of the license is included in the section entitled{ldquo}GNU
   Free Documentation License{rdquo}.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “ with…Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
   Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.