Advanced Topics

Advanced Topics

Publication Date: 2018-10-25

SUSE LLC
10 Canal Park Drive
Suite 200
Cambridge MA 02141
USA
https://www.suse.com/documentation

Contents

About This Guide

Note: Uyuni Version Information

In this manual if not other specified, Uyuni version 3.2 is assumed and this version is required if a feature is discussed. Uyuni 3.2 and Uyuni 3.2 Proxy were originally released as a SLES 12 SP3 extension. Whenever features of the Uyuni3.2 host operating system are documented and not other specified version 12 SP3 is assumed.

1 SUSE Manager on IBM z Systems

1.1 Introduction

This best practice guide is intended for z/VM administrators responsible for operating the IBM z Systems Mainframe. The goal of this guide is to lead an z/VM administrator trained on normal z Systems operating protocols through the installation of Uyuni onto an existing mainframe system. The intent of this article is not to cover the variety of hardware configuration profiles available on z Systems but instead to provide a foundational overview of the procedure and requirements necessary for a successful Uyuni server deployment.

1.2 Base System Requirements

The z/VM administrator should acquire and prepare the following resources for a successful Uyuni installation. Uyuni3.2 is delivered as an extension. These sections will provide you with the minimum and recommended system requirements for Uyuni . The base system for Uyuni 3.2 is SLES 12SP3 .

Hardware	**Recommended Hardware**
IBM Systems	* IBM zEnterprise System z196 *(z196)* * IBM zEnterprise System z114 *(z114)* * IBM zEnterprise EC12 *(zEC12)* * IBM zEnterprise EC12 *(zEC12)* * IBM zEnterprise BC12 *(zBC12)* * IBM z13 *(z13)* * LinuxOne Rockhopper * LinuxOne Emperor
RAM	*Split memory requirements across available RAM, VDISK and swap to suit your environment. On a production system the ratio of physical memory to VDISK will need to be re-evaluated based on the number of clients which will be supported.* Minimum 5 GB+ for test server *(3 GB RAM + 2 GB VDISK Swap)*

Hardware	Recommended Hardware
	Minimum 16 GB+ for base installation Minimum 32 GB+ for a production server
Free Disk Space	Minimum 100 GB for root partition Minimum 50 GB for `/var/lib/pgsql` Minimum 50 GB per SUSE product + 100 GB per Red Hat product `/var/spacewalk`
Network Connection	* OSA Express Ethernet *(including Fast and Gigabit Ethernet) _ * HiperSockets or Guest LAN * 10 GBE, VSWITCH * RoCE _(RDMA over Converged Ethernet)* The following interfaces are still included but no longer supported: * CTC *(or virtual CTC)* * IP network interface for IUCV

MEDIA REQUIREMENTS

- SUSE Linux Enterprise 12SP3 Installation Media for IBM z Systems : https://www.suse.com/products/server/download/

1.3 Additional Requirements

There are a few additional resource requirements you will need to prepare before installing the Uyuni extension on your system. This section overviews these requirements.

The guest z/VM should be provided with a static IP address and hostname as these cannot be easily changed after initial setup. The hostname should contain less than 8 characters.

For more information on Uyuni additional requirements, see https://www.suse.com/documentation/suse-manager-3/book_suma_best_practices/data/mgr_conceptual_overview.html.

You will need to ensure you have sufficient disk storage for Uyuni before running `yast2 susemanagersetup`.

This section explains these requirements in more detail.

Warning: UyuniDefault Volume Groups and Disk Space

By default the file system of Uyuni, including the embedded database and patch directories, reside within the root volume. While adjustments are possible once installation is complete, it is the administrator's responsibility to specify and monitor these adjustments.

If your Uyuni runs out of disk space, this can have a severe impact on its database and file structure. Preparing storage requirements in accordance with this section will aid in preventing these harmful effects. SUSE technical services will not be able to provide support for systems suffering from low disk space conditions as this can have an effect on an entire system and therefore becomes unresolvable. A full recovery is only possible with a previous backup or a new Uyuni installation.

Required Storage Devices. An additional disk is required for database storage. This should be an `zFCP` or `DASD` device as these are preferred for use with `HYPERPAV`. The disk should fulfill the following requirements:

- At least 50 GB for `/var/lib/pgsql`
- At least 50 GB for each SUSE product in `/var/spacewalk`
- At least 100 GB for each Red Hat product in `/var/spacewalk`

Reclaiming Disk Space. If you need to reclaim more disk space, try these suggestions:

- Remove custom channels (you cannot remove official SUSE channels)
- Use the `spacewalk-data-fsck --help` command to compare the spacewalk database to the filesystem and remove entries if either is missing.

1.4 SLES 12SP3 Installation and the Uyuni Extension

This section covers the installation of Uyuni3.2 as an extension to SLES 12SP3 .

For more information on deploying SLES 12SP3 on your hardware, see https://www.suse.com/documentation/sles-12/book_sle_deployment/data/cha_zseries.html .

During installation of SLES 12SP3 select Uyuni as an extension.

After rebooting you will need to set up the additional storage required for `/var/spacewalk` and `/var/lib/pgsql` and swap space using the yast partitioner tool. This step is *required* before running `yast2 susemanagersetup`.

After configuring the storage requirements, having executed a YaST online update and completed a full system reboot, run Uyuni setup to finalize the Uyuni installation on your z Systems mainframe:

```
{prompt.root}yast2 susemanagersetup
```

This completes the installation of Uyuni on your z Systems . For more information on beginning management with Uyuni , see Setup Uyuni with YaST (suma-setup-with-yast-sumasetup.xml#suma-setup-with-yast-sumasetup) ↗ .

2 Uyuni 3.2 Proxy

This chapter explains how to install and set up Uyuni 3.2 Proxy. It also provides notes about migrating a previous proxy to version 3.2.

2.1 Overview

Uyuni 3.2 Proxy is a Uyuni add-on that caches software packages on an internal, central server. The proxy caches patch updates from SUSE or custom RPMs generated by third-party organizations. A proxy allows you to use bandwidth more effectively because client systems connect to the proxy for updates, and the Uyuni server is no longer required to handle all client requests. The proxy also supports transparent custom package deployment.

Uyuni Proxy is an open source (GPLv2) solution that provides the following features:

- Cache software packages within a Squid proxy.
- Client systems see the SUSE Manager Proxy as a Uyuni server instance.
- The SUSE Manager Proxy is registered as a client system with the Uyuni server.

The primary goal of a SUSE Manager Proxy is to improve Uyuni performance by reducing bandwidth requirements and accelerating response time.

2.2 Proxy Installation and Connecting Clients

2.2.1 Requirements

The following section provides SUSE Manager Proxy requirements.

Supported Client Systems. For supported clients and their requirements, see Supported Client Systems.

Hardware Requirements. Hardware requirements highly depend on your usage scenario. When planning proxy environments, consider the amount of data you want to cache on your proxy. If your proxy should be a 1:1 mirror of your Uyuni, the same amount of disk space is required. For specific hardware requirements, see the following table.

Hardware	Required
CPU	Multi-core 64-bit CPU (x86_64).
RAM	Minimum 4 GB for a non-production server
	Minimum 16 GB for a production server
Free Disk Space	Minimum 100 GB for base installation and at least 50 GB for caching per SUSE product and +100 GB per Red Hat product; a resizeable partition strongly recommended.

Tip: Storage for Proxy Data

SUSE recommends storing the squid proxy caching data on a separate disk formatted with the XFS file system.

SSL Certificate Password. For installing the proxy, you need the SSL certificate password entered during the initial installation of Uyuni.

Network Requirements. For additional network requirements, see Additional Requirements.

SUSE Customer Center. For using SUSE Manager Proxy, you need an account at SUSE Customer Center (SCC) where your purchased products and product subscriptions are registered. Make sure you have the following subscriptions:

- One or more subscriptions for SUSE Manager Proxy .
- One or more subscriptions for Uyuni .
- Subscriptions for the products on the client systems you want to register with Uyuni via SUSE Manager Proxy .
- Subscriptions to client entitlements for the client system you want to register with Uyuni via SUSE Manager Proxy .

Network Time Protocol (NTP). The connection to the Web server via Secure Sockets Layer (SSL) requires correct time settings on the server, proxy and clients. For this reason, all systems must use NTP. For more information, see https://www.suse.com/documentation/sles-12/book_sle_admin/data/cha_netz_xntp.html ↗.

Virtual Environments. The following virtual environments are supported:

- http://www.linux-kvm.org/page/Main_Page
- http://www.vmware.com/
- http://www.microsoft.com/en-us/server-cloud/solutions/virtualization.aspx

For running SUSE Manager Proxy in virtual environments, use the following settings for the virtual machine (VM):

- At least 1 GB of RAM
- Bridged network

2.2.2 Installation and Setup

The following section will guide you through the installation and setup procedure.

Important: Registering Proxies

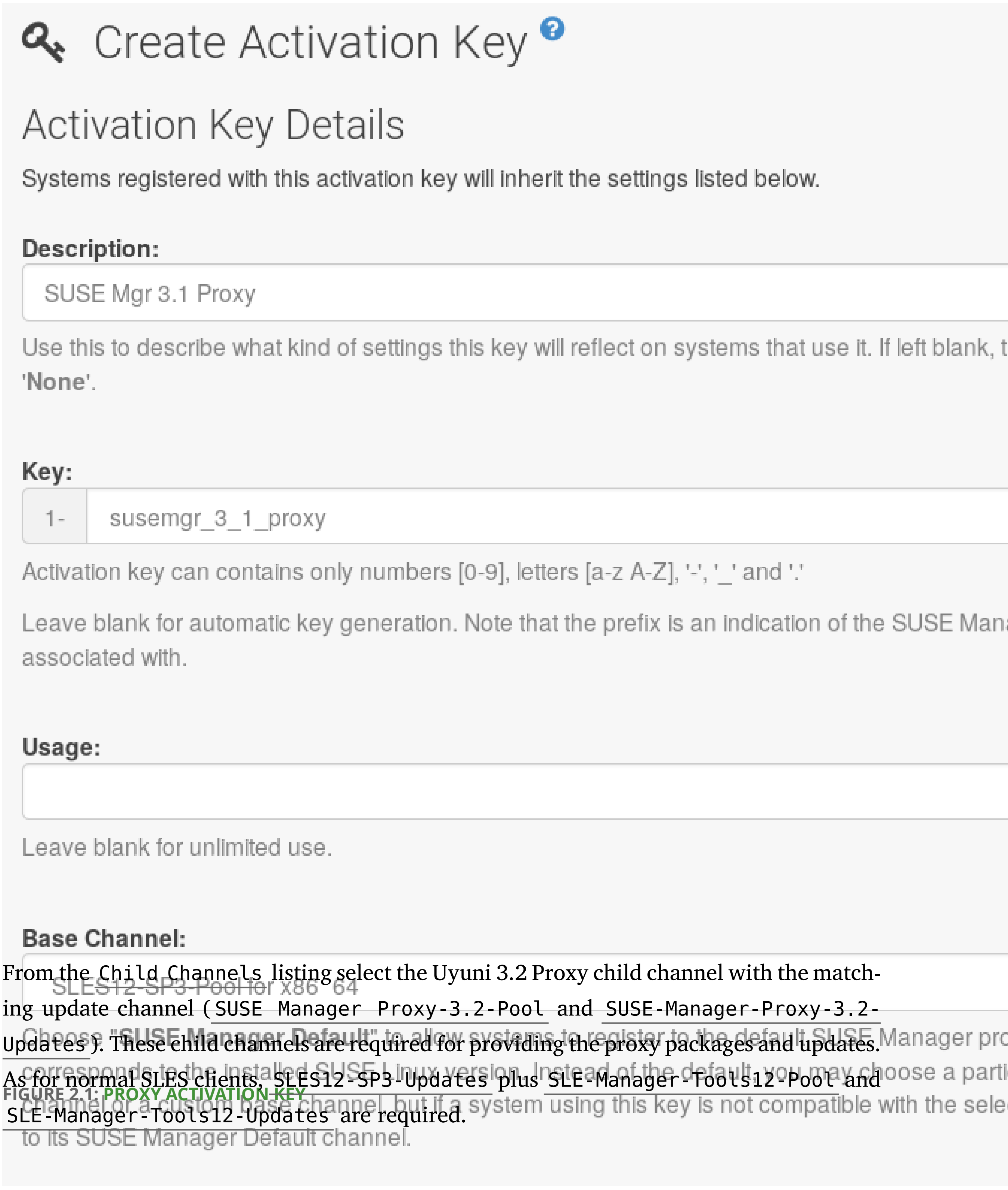
Uyuni Proxy systems are registered as traditional clients or as Salt clients using a bootstrap script. A SUSE Manager Proxy can serve both Traditional and Salt clients.

Important: Procedure: Registering the Proxy

First completely download the channels (SUSE Linux Enterprise 12 SP3) and then create the activation key. Only then you can select the correct child channels.

+

1. Create an activation key based on the SUSE Linux Enterprise 12 SP3 base channel. For more information about activation keys, see .



Create Activation Key

Activation Key Details

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

Description:

SUSE Mgr 3.1 Proxy

Use this to describe what kind of settings this key will reflect on systems that use it. If left blank, t
'None'.

Key:

1- susemgr_3_1_proxy

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

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

Usage:

Leave blank for unlimited use.

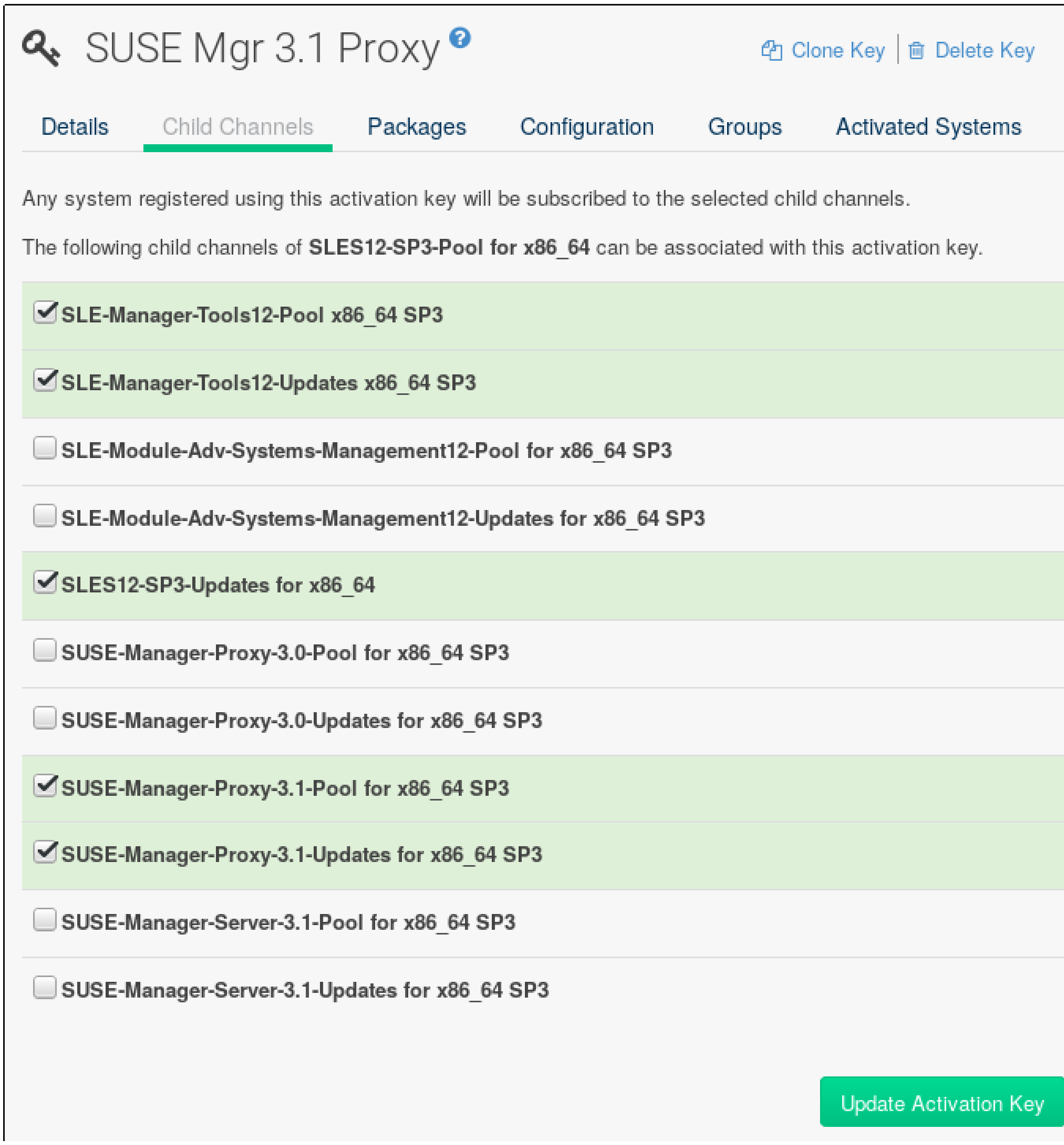
Base Channel:

SLES12-SP3-Pool for x86_64

Choose "SUSE Manager Default" to allow systems to register to the default SUSE Manager pro
corresponds to the installed SUSE Linux version. Instead of the default, you may choose a parti
channel or a custom base channel, but if a system using this key is not compatible with the sele
to its SUSE Manager Default channel.

FIGURE 2.1: PROXY ACTIVATION KEY

2. From the `Child Channels` listing select the Uyuni 3.2 Proxy child channel with the matching update channel (`SUSE Manager Proxy-3.2-Pool` and `SUSE-Manager-Proxy-3.2-Updates`). These child channels are required for providing the proxy packages and updates. As for normal SLES clients, `SLES12-SP3-Updates` plus `SLE-Manager-Tools12-Pool` and `SLE-Manager-Tools12-Updates` are required.



Any system registered using this activation key will be subscribed to the selected child channels.

The following child channels of **SLES12-SP3-Pool for x86_64** can be associated with this activation key.

- [✓] SLE-Manager-Tools12-Pool x86_64 SP3
- [✓] SLE-Manager-Tools12-Updates x86_64 SP3
- [ ] SLE-Module-Adv-Systems-Management12-Pool for x86_64 SP3
- [ ] SLE-Module-Adv-Systems-Management12-Updates for x86_64 SP3
- [✓] SLES12-SP3-Updates for x86_64
- [ ] SUSE-Manager-Proxy-3.0-Pool for x86_64 SP3
- [ ] SUSE-Manager-Proxy-3.0-Updates for x86_64 SP3
- [✓] SUSE-Manager-Proxy-3.1-Pool for x86_64 SP3
- [✓] SUSE-Manager-Proxy-3.1-Updates for x86_64 SP3
- [ ] SUSE-Manager-Server-3.1-Pool for x86_64 SP3
- [ ] SUSE-Manager-Server-3.1-Updates for x86_64 SP3

FIGURE 2.2: **BASE AND CHILD PROXY CHANNEL**

1. Modify a bootstrap script for the proxy. Ensure unchecking *Bootstrap using Salt* , because in this case the proxy must be bootstrapped as a so-called traditional client. For more information about bootstrap scripts, see .
2. Bootstrap the client with the bootstrap script.
3. You will see a list of channels to which your client is already subscribed to. Select the two unchecked proxy channels which include the `SUSE Manager Proxy-3.2-Pool` and `SUSE-Manager-Proxy-3.2-Updates` , then select *Change Subscriptions* to continue. This will provide the required repositories for the proxy packages from the Uyuni server to the client.

A few more steps are still needed:

- install the `suma_proxy` pattern (see *Section 2.2.3, "Install the `suma_proxy` pattern"*);
- copy the SSL certificate and key from the server (see *Section 2.2.4, "Copy Server Certificate and Key"*);
- run `configure-proxy.sh` (see *Section 2.2.5, "Running `configure-proxy.sh`"*);

You will then be able to register your clients against the proxy using the Web UI or a bootstrap script as if it were a Uyuni server. For more information, see *Section 2.2.6, "Registering Salt Clients via SUSE Manager Proxy"*.

2.2.3 Install the `suma_proxy` pattern

Make sure the `suma_proxy` pattern version 2.5.1.3 or later is installed using the following command on the proxy as root:

```
zypper in -t pattern suma_proxy
```

The new salt-broker service will be automatically started at the end of the package installation. This service forwards the Salt interactions to the Uyuni server.

Note: Proxy Chains

It is possible to arrange Salt proxies in a chain. In such a case, the upstream proxy is named "parent".

Make sure the proxie's TCP ports `4505` and `4506` are open and that the proxy can reach the Uyuni server (or another upstream proxy) on these ports.

2.2.4 Copy Server Certificate and Key

The proxy will share some SSL information with the Uyuni server, so the next step is to copy the certificate and its key from the Uyuni server or the upstream proxy.

As root, enter the following commands on the proxy using your Uyuni server or chained proxy named as `PARENT`:

```
cd /root/ssl-build scp root@`PARENT`:/root/ssl-build/RHN-ORG-PRIVATE-SSL-KEY scp
 root@`PARENT`:/root/ssl-build/RHN-ORG-TRUSTED-SSL-CERT scp root@`PARENT`:/root/ssl-
build/rhn-ca-openssl.cnf .
```

Note: Known Limitation

The SUSE Manager Proxy functionality is only supported if the SSL certificate was signed by the same CA as the Uyuni Server certificate. Using certificates signed by different CAs for Proxies and Server is not supported.

2.2.5 Running `configure-proxy.sh`

The `configure-proxy.sh` script will finalize the setup of your SUSE Manager Proxy.

Now execute the interactive `configure-proxy.sh` script. Pressing `Enter` without further input will make the script use the default values provided between brackets `[]`. Here is some information about the requested settings:

Uyuni Parent

A Uyuni parent can be either another proxy server or a Uyuni server.

HTTP Proxy

A HTTP proxy enables your Uyuni proxy to access the Web. This is needed if where direct access to the Web is prohibited by a firewall.

Proxy Version to Activate

Normally, the correct value (3.0, 3.1, or 3.2) should be offered as a default.

Traceback Email

An email address where to report problems.

Use SSL

For safety reasons, press `Y`.

Do You Want to Import Existing Certificates?

Answer `N`. This ensures using the new certificates that were copied previously from the Uyuni server.

Organization

The next questions are about the characteristics to use for the SSL certificate of the proxy. The organization might be the same organization that was used on the server, unless of course your proxy is not in the same organization as your main server.

Organization Unit

The default value here is the proxy's hostname.

City

Further information attached to the proxy's certificate. Beware the country code must be made of two upper case letters. For further information on country codes, refer to the online list of alpha-2 codes (https://www.iso.org/obp/ui/#search).

Tip: Country Code

As the country code enter the country code set during the SUSE Manager installation. For example, if your proxy is in US and your Uyuni in DE, you must enter `DE` for the proxy.

Cname Aliases (Separated by Space)

Use this if your proxy server can be accessed through various DNS CNAME aliases. Otherwise it can be left empty.

CA Password

Enter the password that was used for the certificate of your Uyuni server.

Do You Want to Use an Existing SSH Key for Proxying SSH-Push Salt Minions?

Use this option if you want to reuse a SSH key that was used for SSH-Push Salt minions on the server.

Create and Populate Configuration Channel rhn_proxy_config_1000010001?

Accept default `Y`.

SUSE Manager Username

Use same user name and password as on the Uyuni server.

Activate advertising proxy via SLP?

SLP stands for Service Location Protocol.

If parts are missing, such as CA key and public certificate, the script prints commands that you must execute to integrate the needed files. When the mandatory files are copied, re-run `configure-proxy.sh`. Also restart the script if a HTTP error was met during script execution.

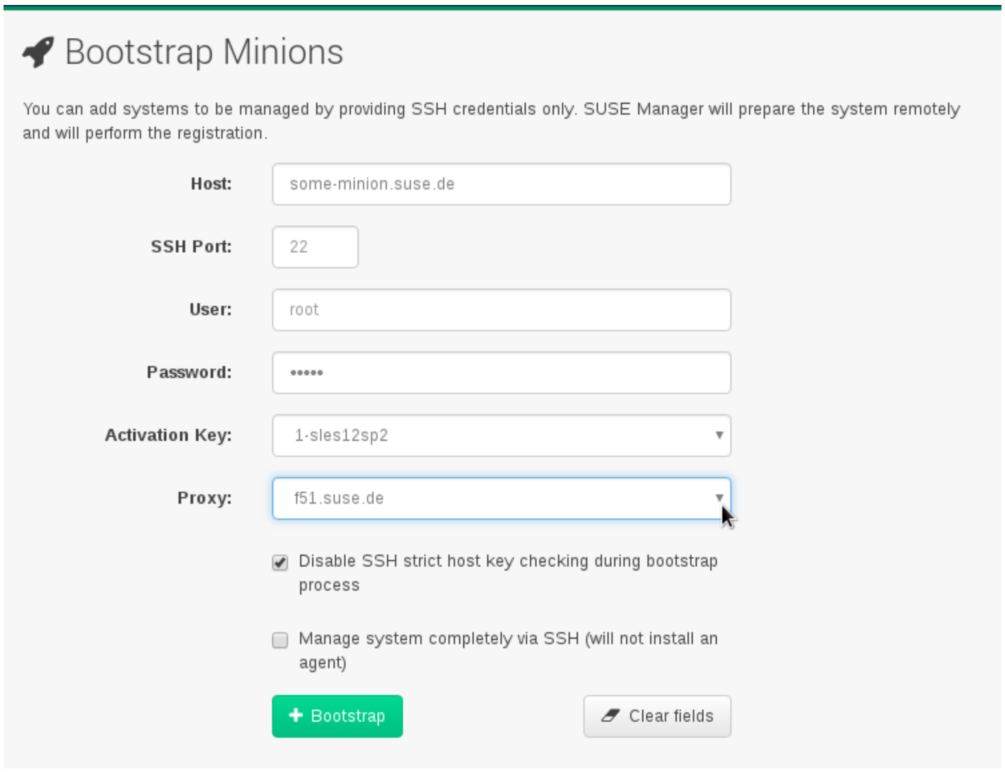
`configure-proxy.sh` activates services required by Uyuni Proxy, such as `squid`, `apache2`, `salt-broker`, and `jabberd`.

To check the status of the proxy system and its clients, click the proxy system's details page on the Web UI (*Systems* › *Proxy*, then the system name). *Connection* › *] and menu:Proxy[* subtabs display the respective status information.

2.2.6 Registering Salt Clients via SUSE Manager Proxy

Proxy servers may now act as a broker and package cache for Salt minions. These minions can be registered with a bootstrap script like the traditional clients, or directly from the Web UI or the command line.

Registering Salt clients via SUSE Manager Proxy from the Web UI is done almost the same way as registering clients directly with the Uyuni server. The difference is that you specify the name of the proxy in the *Proxy* › *] drop-box on menu:Salt[Bootstrapping* page.


Bootstrap Minions

You can add systems to be managed by providing SSH credentials only. SUSE Manager will prepare the system remotely and will perform the registration.

Host:	some-minion.suse.de
SSH Port:	22
User:	root
Password:	•••••
Activation Key:	1-sles12sp2
Proxy:	f51.suse.de

☑ Disable SSH strict host key checking during bootstrap process

☐ Manage system completely via SSH (will not install an agent)

FIGURE 2.3: BOOTSTRAPPING A SALT CLIENT WITH A PROXY

PROCEDURE: REGISTER A SALT CLIENT THROUGH A PROXY FROM COMMAND LINE

1. Instead of the Web UI , you may use the command line to register a minion through a proxy. To do so, 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 proxy, 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.

2.2.7 Registering Clients via SUSE Manager Proxy with a Script

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 refers some parameters (such as activation keys or GPG keys) that depend on your specific setup.

1. Create a client activation key on the Uyuni server using the Web UI. See .
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. An important option is `--traditional` that enables to opt for a traditional client instead of a salt minion. To view available options type `mgr-bootstrap --help` from the command line:

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

3. Optionally edit the resulting bootstrap script. Execute the bootstrap script on the clients as described in .

The clients are registered with the SUSE Manager Proxy specified in the bootstrap script.

2.2.8 Additional Information about Client Registration on Proxies

Within the Web UI, standard proxy pages will show information about client, no matter whether minions or traditional clients.

A list of clients connected to a proxy can be located under *Systems ›] <proxy name › menu:Details[Proxy*.

A list of chained proxies for a minion can be located under *Systems ›] <minion name › menu:Details[Connection*

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

2.3 Enabling PXE Boot via SUSE Manager Proxy

2.3.1 Synchronizing Profiles and System Information

To enable PXE boot via a proxy server, additional software must be installed and configured on both the Uyuni server and the SUSE Manager Proxy server.

1. On the Uyuni server install susemanager-tftpsync :

   ```
   zypper in susemanager-tftpsync
   ```

2. On the SUSE Manager Proxy server install susemanager-tftpsync-recv :

   ```
   zypper in susemanager-tftpsync-recv
   ```

3. Run the `configure-tftpsync.sh` setup script and enter the requested information:

   ```
   configure-tftpsync.sh
   ```

   It asks for hostname and IP address of the Uyuni server and of the proxy itself. Additionally, it asks for the tftpboot directory on the proxy.

4. On the Uyuni server, run `configure-tftpsync.sh` to configure the upload to the SUSE Manager Proxy server:

   ```
   configure-tftpsync.sh FQDN_of_Proxy_Server
   ```

5. To initiate an initial synchronization on the SUSE Manager Server run:

   ```
   cobbler sync
   ```

   Also can also be done after each a change within Cobbler that needs to be synchronized immediately. Otherwise Cobbler synchronization will also run automatically when needed. For more information about Cobbler, see *Chapter 10, Cobbler*.

2.3.2 Configuring DHCP for PXE via SUSE Manager Proxy

Uyuni is using Cobbler to provide provisioning. PXE (tftp) is installed and activated by default. To enable systems to find the PXE boot on the SUSE Manager Proxy server add the following to the DHCP configuration for the zone containing the systems to be provisioned:

```
next-server:`IP_Address_of_SUSE_Manager_Proxy_Server`filename: "pxelinux.0"
```

2.4 Migrating Uyuni 3.1 Proxy to Version 3.2 [Management]

The recommended order for migrations is to first migrate the server and then the proxies.

For the migration of traditionally managed proxies there are two possible approaches:

- Existing Uyuni proxies may be upgraded to version 3.2 with YaST or `zypper` migration.
- Alternatively, the proxies may be replaced by new ones.

This section documents both approaches.

Note: Migrating Uyuni 3 Proxy and Earlier

For migrating Uyuni 3 Proxy and earlier, see https://www.suse.com/documentation/suse-manager-3/book_suma_advanced_topics_31/data/sect1_chapter_book_suma_advanced_topics_31.html, Chapter "SUSE Manager 3.1 Proxy".

2.4.1 Replacing a SUSE Manager Proxy

A SUSE Manager Proxy is `dumb` in the sense that it does not contain any information about the clients which are connected to it. A SUSE Manager Proxy can therefore be replaced by a new one. Naturally, the replacement proxy must have the same name and IP address as its predecessor.

In order to replace a SUSE Manager Proxy and keeping the clients registered to the proxy leave the old proxy in Uyuni. Create a reactivation key for this system and then register the new proxy using the reactivation key. If you do not use the reactivation key, you will need to re-registered all the clients against the new proxy.

PROCEDURE: REPLACING A SUSE MANAGER PROXYAND KEEPING THE CLIENTSREGISTERED

1. Before starting the actual migration procedure, save the data from the old proxy, if needed. Consider copying important data to a central place that can also be accessed by the new server:
   - Copy the scripts that are still needed.
   - Copy the activation keys from the previous server. Of course, it is always better to re-create the keys.
2. Shutdown the server.
3. Install a new Uyuni 3.2 Proxy, see *Section 2.2, “Proxy Installation and Connecting Clients”*.
4. In the Uyuni Web UI select the newly installed SUSE Manager Proxy and delete it from the systems list.
5. In the Web UI, create a reactivation key for the old proxy system: On the System Details tab of the old proxy click `Reactivation`. Then click `Generate New Key`, and remember it (write it on a piece of paper or copy it to the clipboard). For more information about reactivation keys, see .
6. After the installation of the new proxy, perform the following actions (if needed):
   - Copy the centrally saved data to the new proxy system.
   - Install any other needed software.
   - If the proxy is also used for autoinstallation, do not forget to setup TFTP synchronization.

Important: Proxy Installation and Client Connections

During the installation of the proxy, clients will not be able to reach the Uyuni server. After a SUSE Manager Proxy system has been deleted from the systems list, all clients connected to this proxy will be (incorrectly) listed as `directly connected` to the Uyuni server. After the first successful operation on a client *such as execution of a remote command or installation of a package or patch* this information will automatically be corrected. This may take a few hours.

2.4.2 Upgrading a SUSE Manager Proxy from 3.1 to 3.2

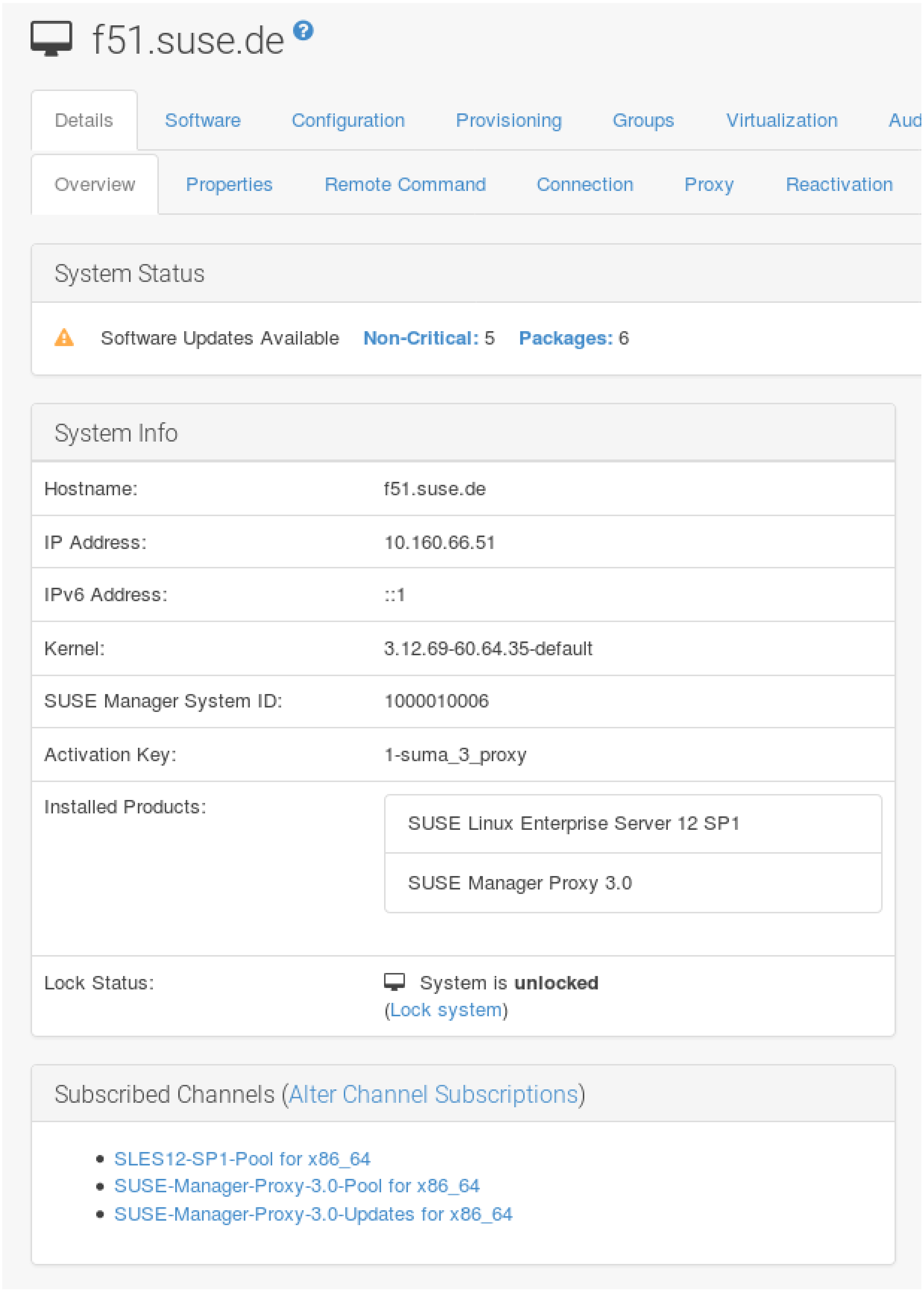
In most situations upgrading the proxy will be your preferred solution as this retains all cached packages. Selecting this route saves time especially regarding proxies connected to Uyuni server via low-bandwith links. This upgrade is similar to a standard client migration.

Warning: Synchronizing Target Channels

Before successfully initializing the product migration, you first must make sure that the migration target channels are completely mirrored. For the upgrade to Uyuni 3.2 Proxy, at least the `SUSE Linux Enterprise Server 12 SP3` base channel with the `SUSE Manager Proxy 3.2` child channel for your architecture is required.

PROCEDURE: MIGRATING PROXY TO 3.2

1. Direct your browser to the Uyuni Web UI where your proxy is registered, and login.
2. On the *Main Menu* › *Systems* › *Systems* › *Proxy* page select your proxy server from the table.



f51.suse.de

System Status

Software Updates Available **Non-Critical:** 5 **Packages:** 6

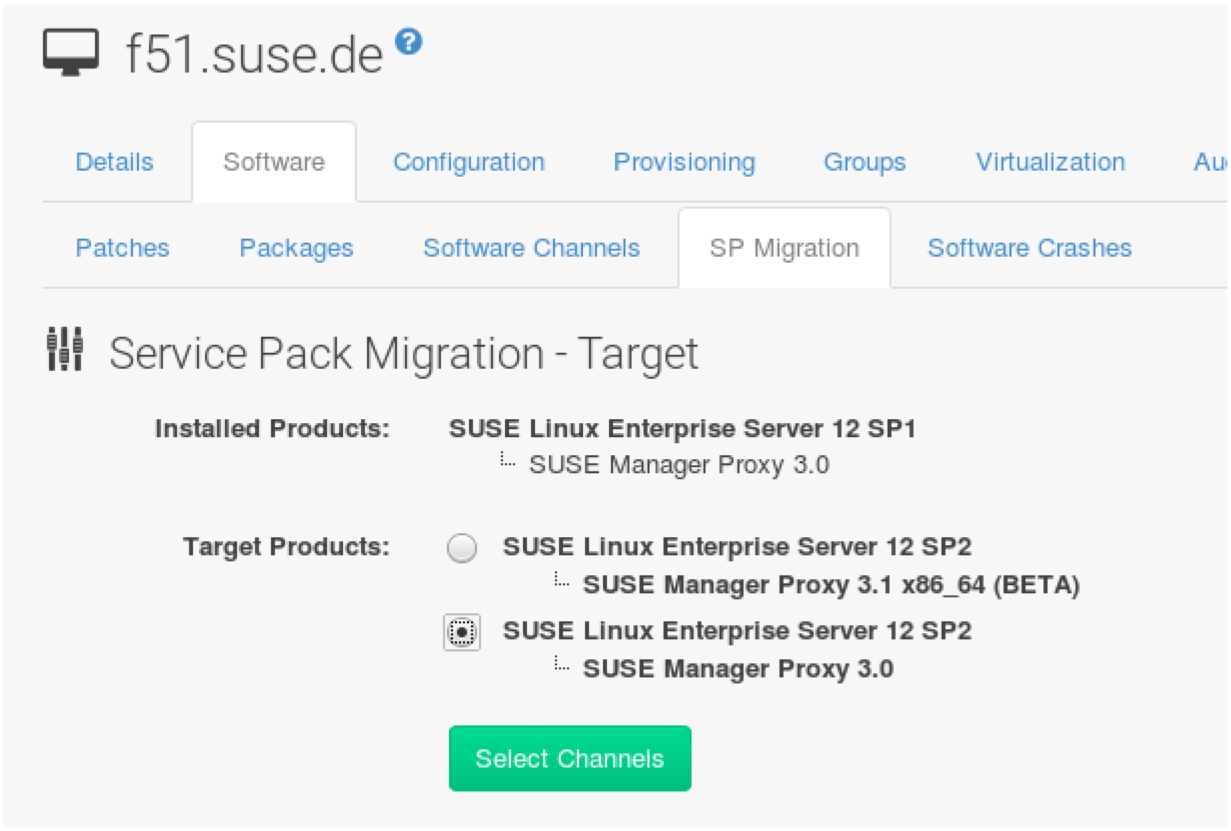
System Info

Hostname:	f51.suse.de
IP Address:	10.160.66.51
IPv6 Address:	::1
Kernel:	3.12.69-60.64.35-default
SUSE Manager System ID:	1000010006
Activation Key:	1-suma_3_proxy
Installed Products:	SUSE Linux Enterprise Server 12 SP1 SUSE Manager Proxy 3.0
Lock Status:	System is **unlocked** (Lock system)

Subscribed Channels (Alter Channel Subscriptions)

- SLES12-SP1-Pool for x86_64
- SUSE-Manager-Proxy-3.0-Pool for x86_64
- SUSE-Manager-Proxy-3.0-Updates for x86_64

3. On the system's detail page select the *Software* › *SP Migration* tab.



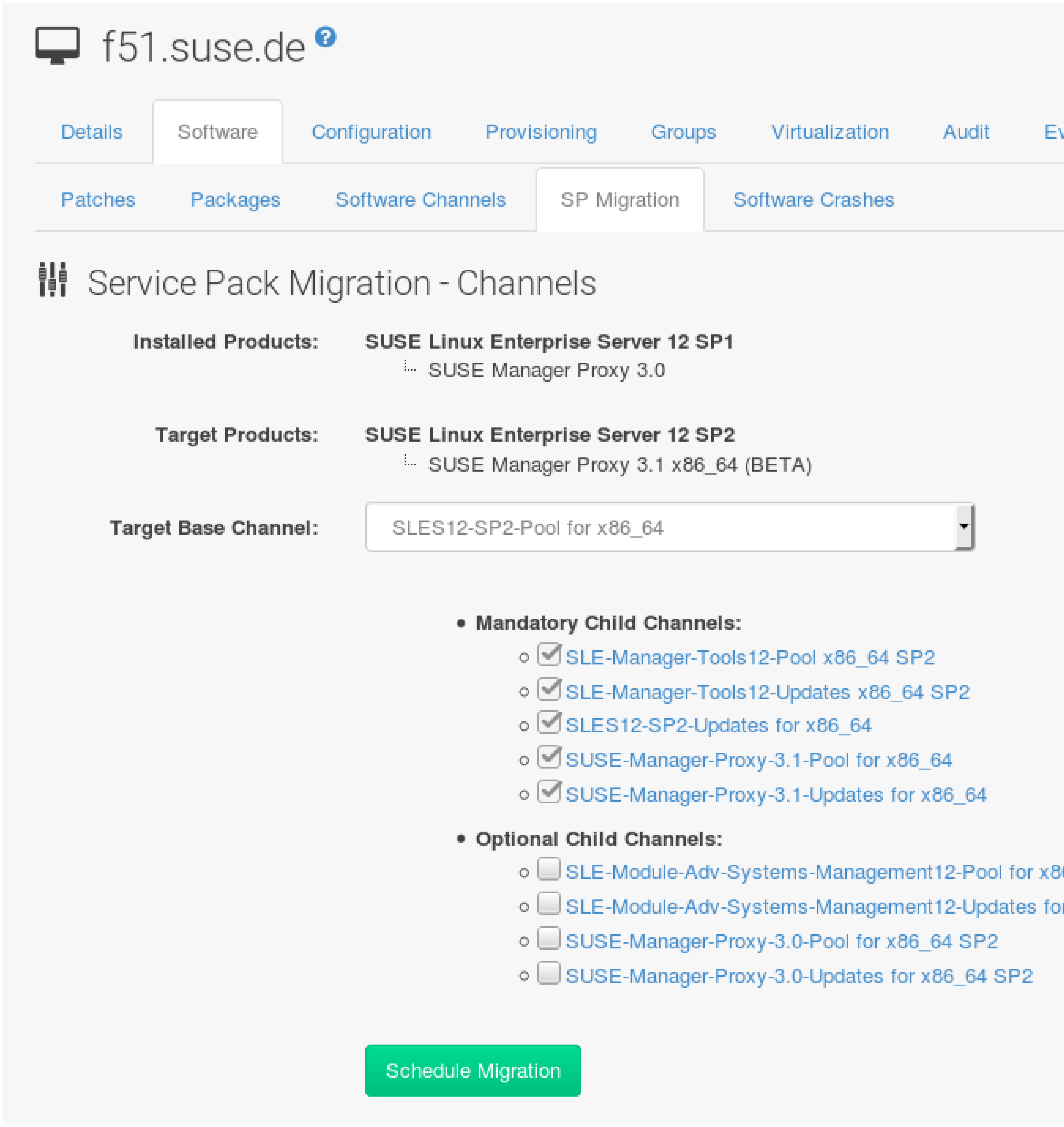
Service Pack Migration - Target

Installed Products: **SUSE Linux Enterprise Server 12 SP1**
└ SUSE Manager Proxy 3.0

Target Products:
( ) **SUSE Linux Enterprise Server 12 SP2**
└ **SUSE Manager Proxy 3.1 x86_64 (BETA)**
(●) **SUSE Linux Enterprise Server 12 SP2**
└ **SUSE Manager Proxy 3.0**

4. From this page you will see installed products listed on your proxy client, and the available target products. Select the wanted `Target Products`, in this case `SUSE Linux Enterprise Server 12 SP3` with `SUSE Manager Proxy 3.2`.

5. Then confirm with *Select Channels*.



f51.suse.de

Service Pack Migration - Channels

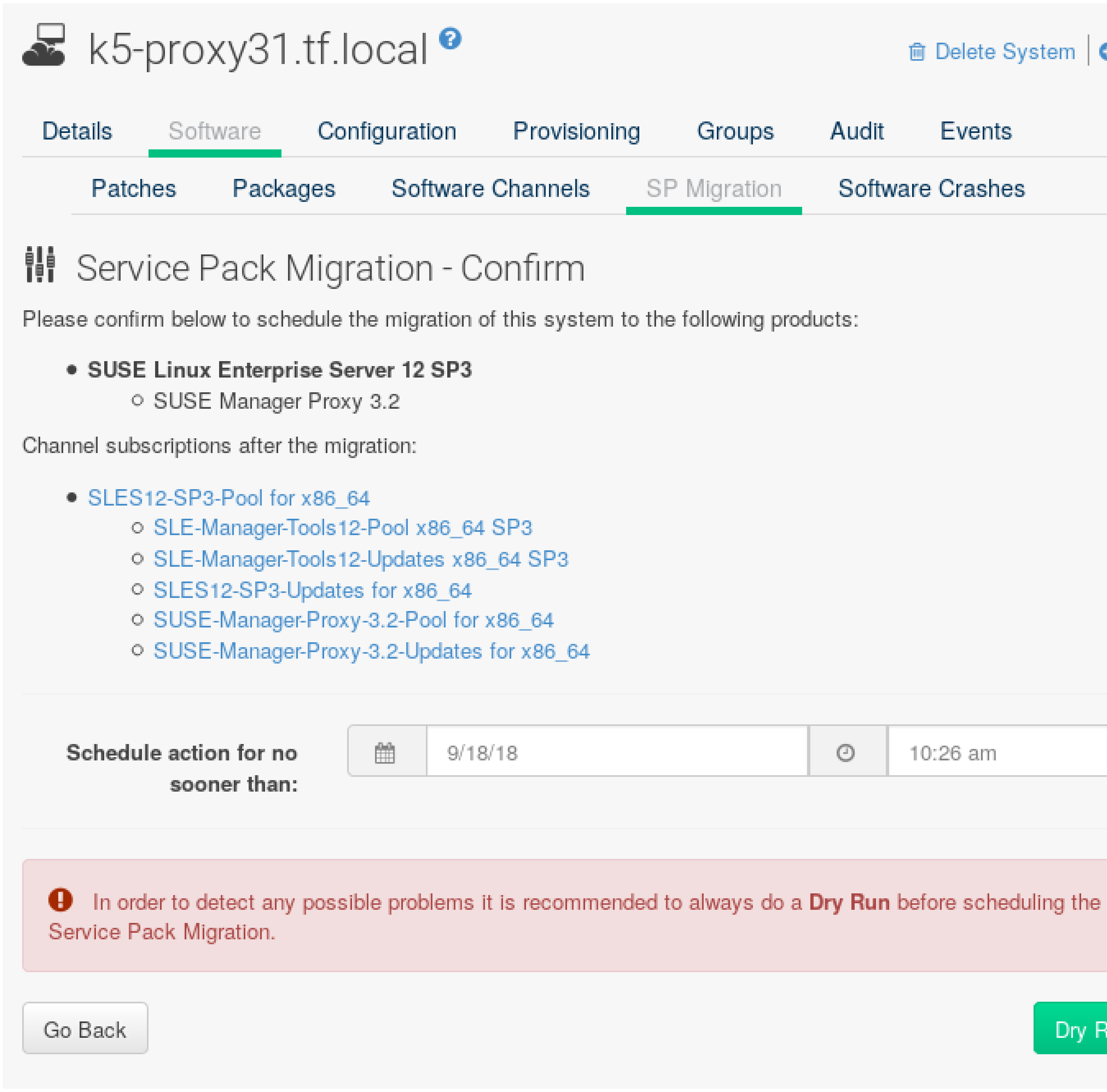
Installed Products: **SUSE Linux Enterprise Server 12 SP1**
SUSE Manager Proxy 3.0

Target Products: **SUSE Linux Enterprise Server 12 SP2**
SUSE Manager Proxy 3.1 x86_64 (BETA)

Target Base Channel: SLES12-SP2-Pool for x86_64

- **Mandatory Child Channels:**
  - ✓ SLE-Manager-Tools12-Pool x86_64 SP2
  - ✓ SLE-Manager-Tools12-Updates x86_64 SP2
  - ✓ SLES12-SP2-Updates for x86_64
  - ✓ SUSE-Manager-Proxy-3.1-Pool for x86_64
  - ✓ SUSE-Manager-Proxy-3.1-Updates for x86_64
- **Optional Child Channels:**
  - ☐ SLE-Module-Adv-Systems-Management12-Pool for x8
  - ☐ SLE-Module-Adv-Systems-Management12-Updates fo
  - ☐ SUSE-Manager-Proxy-3.0-Pool for x86_64 SP2
  - ☐ SUSE-Manager-Proxy-3.0-Updates for x86_64 SP2

6. From the `Schedule Migration` menu, select the time and click *Confirm*.


k5-proxy31.tf.local

Delete System

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

Patches | Packages | Software Channels | SP Migration | Software Crashes

Service Pack Migration - Confirm

Please confirm below to schedule the migration of this system to the following products:

- **SUSE Linux Enterprise Server 12 SP3**
  - SUSE Manager Proxy 3.2

Channel subscriptions after the migration:

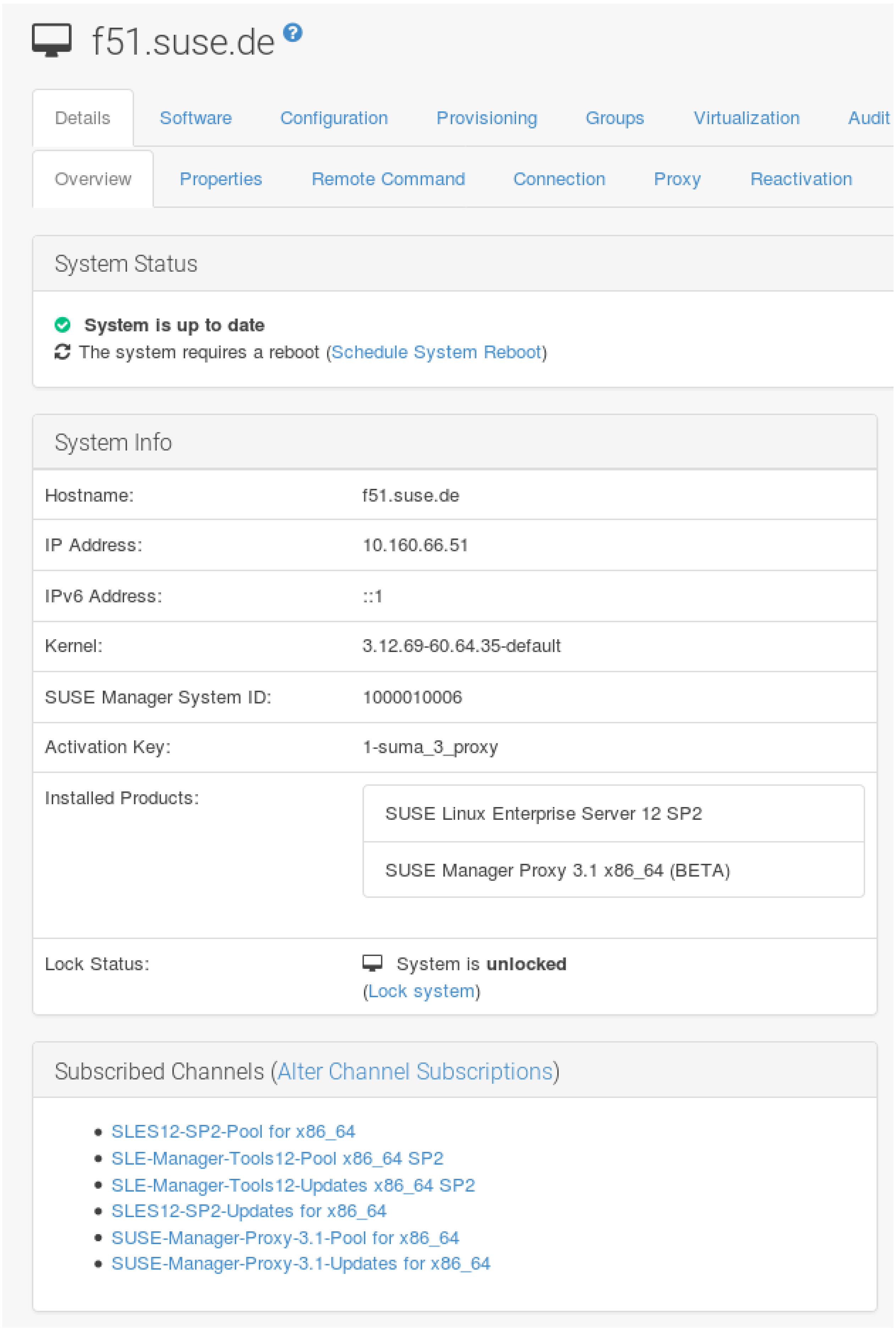
- SLES12-SP3-Pool for x86_64
  - SLE-Manager-Tools12-Pool x86_64 SP3
  - SLE-Manager-Tools12-Updates x86_64 SP3
  - SLES12-SP3-Updates for x86_64
  - SUSE-Manager-Proxy-3.2-Pool for x86_64
  - SUSE-Manager-Proxy-3.2-Updates for x86_64

Schedule action for no sooner than: 9/18/18 10:26 am

In order to detect any possible problems it is recommended to always do a **Dry Run** before scheduling the Service Pack Migration.

Go Back | Dry R

Check the `System Status` on the *System Details* › *Overview* when the migration is done.


f51.suse.de

System Status

System is up to date

The system requires a reboot (Schedule System Reboot)

System Info

Hostname:	f51.suse.de
IP Address:	10.160.66.51
IPv6 Address:	::1
Kernel:	3.12.69-60.64.35-default
SUSE Manager System ID:	1000010006
Activation Key:	1-suma_3_proxy
Installed Products:	SUSE Linux Enterprise Server 12 SP2 SUSE Manager Proxy 3.1 x86_64 (BETA)
Lock Status:	System is **unlocked** (Lock system)

Subscribed Channels (Alter Channel Subscriptions)

- SLES12-SP2-Pool for x86_64
- SLE-Manager-Tools12-Pool x86_64 SP2
- SLE-Manager-Tools12-Updates x86_64 SP2
- SLES12-SP2-Updates for x86_64
- SUSE-Manager-Proxy-3.1-Pool for x86_64
- SUSE-Manager-Proxy-3.1-Updates for x86_64

Finally consider scheduling a reboot.

3 Security

3.1 Setup a Minion to Master Validation Fingerprint

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

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

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

   On your minion, open the minion configuration file located in `/etc/salt/minion`. Uncomment the following line and enter the masters fingerprint replacing the example fingerprint:

   ```
   master_finger: 'ba:30:65:2a:d6:9e:20:4f:d8:b2:f3:a7:d4:65:11:13'
   ```

2. Restart the salt-minion service:

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

For more information on configuring security from a minion see: https://docs.saltstack.com/en/latest/ref/configuration/minion.html

3.2 Signing Repository Metadata

3.2.1 Generate a Custom GPG key

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

```
$> gpg --gen-key

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
```

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

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

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

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

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

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

3.2.2 Configure signing metadata

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

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

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

```
KEYID="607FABDB"

GPGPASS="MySecretPassword"
```

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

```
sign_metadata = 1
```

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

3.2.3 Regenerate all metadata

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

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

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

3.2.4 Trust the GPG key on all clients

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

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

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

2. Import the GPG key on all clients
   This can be done using a remote command on all clients

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

Tip: Tip

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

4 Uyuni in the Public Cloud: Uyuni Server and Uyuni Proxy in the Public Cloud

Uyuni delivers best-in-class Linux server management capabilities. For detailed information about the product please refer to the SUSEManager (https://www.suse.com/documentation/suse_manager) ↗ documentation.

The Uyuni Server and Uyuni Proxy images published by SUSE in selected Public Cloud environments are provided as Bring Your Own Subscription (BYOS) images. Uyuni Server instances need to be registered with the SUSE Customer Center (SCC). Subscriptions of Uyuni Proxy instances are handled through their parent Uyuni Server. After an instance is launched, Uyuni needs to be set up and configured following the procedure in the Uyuni documentation.

4.1 Instance Requirements

Select an instance size that meets the system requirements as documented in the Uyuni documentation.

- Minimal main memory: >12G
- The Uyuni setup procedure performs a Forward-confirmed reverse DNS lookup. This must succeed in order for the setup procedure to complete successfully and for Uyuni to operate as expected. Therefore it is important that the hostname and IP configuration be performed prior to running the Uyuni setup procedure.
- Uyuni Server and Uyuni Proxy instances are expected to run in a network configuration that provides you control over DNS entries and that is shielded from the Internet at large. Within this network configuration DNS (Domain Name Service) resolution must be provided, such that `hostname -f` returns the FQDN (Full Qualified Domain Name). The DNS resolution is not only important for the Uyuni Server procedure but is also important when clients are configured to be managed via Uyuni. Configuring DNS is Cloud Framework dependent, please refer to the cloud service provider documentation for detailed instructions.
- Minimal free disk space for Uyuni 15G.
  For Public Cloud instances we recommend that the repositories and the Uyuni Server database, and respectively the Uyuni Proxy squid cache, be located on an external virtual disk. The details for this setup are provided in *Section 4.2.1, "Using Separate Storage Volume"*.

Storing the database and the repositories on an external virtual disk ensures that the data is not lost should the instance need to be terminated for some reason.

Please ensure that the selected instance type matches the requirements listed above. Although we recommend that the database and the repositories are stored on a separate device it is still recommended to set the root volume size of the instance to 20G.

4.2 Setup

Run an instance of the Uyuni Server or Uyuni Proxy image as published by SUSE. The images are identifiable by the suse, manager, server or proxy, and byos keywords in each public cloud environment. The Uyuni instance must run in a network access restricted environment such as the private subnet of a VPC or with an appropriate firewall setting such that it can only be accessed by machines in the IP ranges you use. A generally accessible Uyuni instance violates the terms of the Uyuni EULA. Access to the web interface of SUSE Manager requires https.

Uyuni requires a stable and reliable hostname, changing the hostname can create errors. In this procedure, all commands need to be executed as the root user.

PROCEDURE: SETTING THE HOSTNAME

1. Disable hostname setup by editing the DHCP configuration file at `/etc/sysconfig/network/dhcp`, and adding this line:

```
DHCLIENT_SET_HOSTNAME="no"
```

2. Set the hostname locally using the `hostnamectl` command. Ensure you use the system name, not the fully qualified domain name (FQDN). The FQDN is set by the cloud framework; for example if **cloud_instance.cloud.net** is the FQDN, **cloud_instance** is the system name and **cloud.net** is the domain name:

```
hostnamectl set-hostname system_name
```

3. Create a DNS entry in your network environment for domain name resolution, or force correct resolution by editing the `/etc/hosts` file:

```
$ echo "${local_address} suma.cloud.net suma" >> /etc/hosts
```

You can locate the local address by checking your public cloud web console, or from the command line using these commands:

- Amazon EC2 instance:

```
$ ec2metadata --local-ipv4
```

- Google Compute Engine:

```
$ gcemetadata --query instance --network-interfaces --ip
```

- Microsoft Azure:

```
$ azuremetadata --internal-ip
```

Forcing DNS resolution by modifying the `/etc/hosts` file will work correctly in most environments. However, you will have to perform the same modification for any client that is to be managed with this Uyuni instance. In most cases, it will be easier to manage the DNS resolution by creating a DNS entry in your network environment.

One other method for managing hostname resolution is by editing the `/etc/resolv.conf` file. Depending on the order of your setup, for example if you started the Uyuni instance prior to setting up DNS services the file may not contain the appropriate **search** directive. Double check that the proper search directive exists in `/etc/resolv.conf`. In our example the directive would be **search cloud.net**. If the directive is missing add it to the file.

For an update of the DNS records for the instance within the DNS service of your network environment, refer to the cloud service provider documentation for detailed instructions: * **DNS setup on Amazon EC2 (http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-dns.html)** ↗ * DNS setup on Google Compute Engine (https://cloud.google.com/compute/docs/networking) ↗ * DNS setup on Microsoft Azure (https://azure.microsoft.com/en-us/documentation/articles/dns-operations-recordsets) ↗

If you run a Uyuni Server instance, you can run YaST after the instance is launched to ensure the external storage is attached and prepared according to *Section 4.2.1, "Using Separate Storage Volume"*, and the DNS resolution is set up as described:

```
$ /sbin/yast2 susemanager_setup
```

Note that the setup of Uyuni from this point forward does not differ from the documentation in the SUSE Manager Guide (https://www.suse.com/documentation/suse_manager) ↗.

The Uyuni setup procedure in YaST is designed as a one pass process with no rollback or cleanup capability. Therefore, if the setup procedure is interrupted or ends with an error, it is not recommended that you repeat the setup process or attempts to manually fix the configuration. These methods are likely to result in a faulty Uyuni installation. If you experience errors during setup, start a new instance, and begin the setup procedure again on a clean system.

If you receive a message that there is not enough space available for setup, ensure that your root volume is at least 20GB and double check that the instructions in *Section 4.2.1, "Using Separate Storage Volume"* have been completed correctly.

Uyuni Server for the Public Cloud comes with a bootstrap data module pre-installed. The bootstrap module contains optimized package lists for bootstrapping instances started from SUSE Linux Enterprise images published by SUSE. If you intend to register such an instance, when you creatr the bootstrap repository run the `mgr-create-bootstrap-repo` script using this command, to create a bootstrap repository suitable for SUSE Linux Enterprise 12 SP1 instances.

```
$ mgr-create-bootstrap-repo --datamodule=mgr_pubcloud_bootstrap_data -c SLE-12-SP1-x86_64
```

See Creating the SUSE Manager Tools Repository (https://www.suse.com/documentation/suse-manager-3/book.suma.getting-started/data/create_tools_repository.html) for more information on bootstrapping.

Prior to registering instances started from on demand images remove the following packages from the instance to be registered: … cloud-regionsrv-client … **For Amazon EC2**

+ regionServiceClientConfigEC2

+ regionServiceCertsEC2 … **For Google Compute Engine**

+ cloud-regionsrv-client-plugin-gce

+ regionServiceClientConfigGCE

+ regionServiceCertsGCE … **For Microsoft Azure**

+ regionServiceClientConfigAzure

+ regionServiceCertsAzure

+ If these packages are not removed it is possible to create interference between the repositories provided by Uyuni and the repositories provided by the SUSE operated update infrastructure.

+ Additionally remove the line from the `/etc/hosts` file that contains the **susecloud.net** reference. ** If you run a Uyuni Proxy instance

+ Launch the instance, optionally with external storage configured. If you use external storage (recommended), prepare it according to *Section 4.2.1, "Using Separate Storage Volume"*. It is recommended but not required to prepare the storage before configuring Uyuni proxy, as the suma-storage script will migrate any existing cached data to the external storage. After preparing the instance, register the system with the parent SUSE Manager, which could be a Uyuni Server or another Uyuni Proxy. See the SUSE Manager Proxy Setup guide (https://www.suse.com/documentation/suse-manager-3/singlehtml/suse_manager21/book_susemanager_proxyquick/book_susemanager_proxyquick.html) for details. Once registered, run

+

```
$ /usr/sbin/configure-proxy.sh
```

+ to configure your Uyuni Proxy instance. . After the completion of the configuration step, Uyuni should be functional and running. For Uyuni Server, the setup process created an administrator user with following user name:

+ * User name: `admin`

+

TABLE 4.1: **ACCOUNT CREDENTIALS FOR ADMIN USER**

Amazon EC2	Google Compute Engine	Microsoft Azure
`Instance-ID`	`Instance-ID`	`Instance-Name`**-suma**

+ The current value for the `Instance-ID` or `Instance-Name` in case of the Azure Cloud, can be obtained from the public cloud Web console or from within a terminal session as follows: ** Obtain instance id from within Amazon EC2 instance

+

```
$ ec2metadata --instance-id
```

- Obtain instance id from within Google Compute Engine instance

  ```
  $ gcemetadata --query instance --id
  ```

- Obtain instance name from within Microsoft Azure instance

  ```
  $ azuremetadata --instance-name
  ```

  After logging in through the Uyuni Server Web UI, **change** the default password.

Uyuni Proxy does not have administration access to the Web UI. It can be managed through its parent Uyuni Server.

4.2.1 Using Separate Storage Volume

We recommend that the repositories and the database for Uyuni be stored on a virtual storage device. This best practice will avoid data loss in cases where the Uyuni instance may need to be terminated. These steps **must** be performed **prior** to running the YaST Uyuni setup procedure.

1. Provision a disk device in the public cloud environment, refer to the cloud service provider documentation for detailed instructions. The size of the disk is dependent on the number of distributions and channels you intend to manage with Uyuni. For sizing information refer to SUSE Manager sizing examples (https://www.suse.com/support/kb/doc.php?id=7015050). A rule of thumb is 25 GB per distribution per channel.

2. Once attached the device appears as Unix device node in your instance. For the following command to work this device node name is required. In many cases the attached storage appears as **/dev/sdb**. In order to check which disk devices exists on your system, call the following command:

```
$ hwinfo --disk | grep -E "Device File:"
```

3. With the device name at hand the process of re-linking the directories in the filesystem Uyuni uses to store data is handled by the suma-storage script. In the following example we use `/dev/sdb` as the device name.

```
$ /usr/bin/suma-storage /dev/sdb
```

   After the call all database and repository files used by SUSE Manager Server are moved to the newly created xfs based storage. In case your instance is a Uyuni Proxy, the script will move the Squid cache, which caches the software packages, to the newly created storage. The xfs partition is mounted below the path `/manager_storage`. .

4. Create an entry in /etc/fstab (optional)
   Different cloud frameworks treat the attachment of external storage devices differently at instance boot time. Please refer to the cloud environment documentation for guidance about the fstab entry.
   If your cloud framework recommends to add an fstab entry, add the following line to the **/etc/fstab** file.

```
/dev/sdb1 /manager_storage xfs defaults,nofail 1 1
```

4.3 Registration of Cloned Systems

Uyuni cannot distinguish between different instances that use the same system ID. If you register a second instance with the same system ID as a previous instance, Uyuni will overwrite the original system data with the new system data. This can occur when you launch multiple instances from the same image, or when an image is created from a running instance. However, it is possible to clone systems and register them successfully by deleting the cloned system's ID, and generating a new ID.

PROCEDURE: REGISTERING CLONED SYSTEMS

1. Clone the system using your preferred hypervisor's cloning mechanism.
2. On the cloned system, change the hostname and IP addresses, and check the `/etc/hosts` file to ensure you have the right host entries.
3. Stop the `rhnsd` daemon with `/etc/init.d/rhnsd stop` or `rcosad stop`.
4. For SLES 11 or Red Hat Enterprise Linux 5 or 6 clients, run these commands:

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

5. For SLES 12 or Red Hat Enterprise Linux 7 clients, run these commands:

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

6. If you are using Salt, then you will also need to run these commands:

```
# service salt-minion stop
# rm -rf /var/cache/salt
```

7. If you are using a traditional registered system, clean up the working files with this command:

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

The bootstrap should now run with a new system ID, rather than a duplicate.

If you are onboarding Salt minion clones, then you will also need to check if they have the same Salt minion ID. You will need to delete the minion ID on each cloned minion, using the `rm` command. Each operating system type stores this file in a slightly different location, check the table for the appropriate command.

Minion ID File Location. Each operating system stores the minion ID file in a slightly different location, check the table for the appropriate command.

Operating System	Commands

SLES 12	`rm /etc/salt/minion_id`
	`rm -f /etc/zypp/credentials.d/{SCC-credentials,NCCcredentials}`

SLES 11	`rm /etc/salt/minion_id`
	`suse_register -E`

SLES 10	`rm -rf /etc/{zmd,zypp}`
	`rm -rf /var/lib/zypp/` Do not delete `/var/lib/zypp/db/products/`
	`rm -rf /var/lib/zmd/`

Red Hat Enterprise Linux 5, 6, 7	`rm -f /etc/NCCcredentials`

Once you have deleted the minion ID file, re-run the bootstrap script, and restart the minion to see the cloned system in Uyuni with the new ID.

5 Optimization and Scalability

5.1 Optimizing Apache and Tomcat

Warning: Altering Apache and Tomcat Parameters

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

5.1.1 Apache's httpd MaxClients Parameter

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

Warning

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

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

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

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

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

5.1.2 Tomcat's maxThreads Parameter

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

Example relevant lines in `server.xml`:

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

Note: Tuning Notes

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

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

Where:

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

5.2 Big Scale Deployment (1000 Minions or More)

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

5.2.1 General Recommendations

SUSE recommends the following in a big scale Uyuni deployment:

- Uyuni servers should have at least 8 recent x86 cores, 32 GiB of RAM, and, most important, fast I/O devices such as at least an SSD (2 SSDs in RAID-0 are strongly recommended).
- Proxies with many minions (hundreds) should have at least 2 recent x86 cores and 16 GiB of RAM.

- Use one SUSE Manager Proxy per 500-1000 clients. Keep into account that download time depends on network capacity. Here is a rough example calculation with physical link speed of 1 GB/s:

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

  This is:

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

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

  ```
  kernel: neighbour table overflow
  ```

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

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

Tip: Start Small and Scale Up

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

5.2.2 Tuning Proposals

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

- Increase the maximum Tomcat heap memory to face a potentially long queue of Salt return results. Set 8 GiB instead of the current default 1 GiB: parameter `Xmx1G` in `/etc/sysconfig/tomcat` (affects onboarding and Action execution).
- Increase the number of Taskomatic workers, allowing to parallelize work on a high number of separate jobs. Set parameter `org.quartz.threadPool.threadCount = 100` in `/etc/rhn/rhn.conf` (affects onboarding and staging).
- Allow Taskomatic to check for runnable jobs more frequently to reduce latency. Set parameter `org.quartz.scheduler.idleWaitTime = 1000` in `/etc/rhn/rhn.conf` (affects onboarding, staging and Action execution).
- Increase Tomcat's Salt return result workers to allow parallelizing work on a high number of Salt return results. Set parameter `java.message_queue_thread_pool_size = 100` in `/etc/rhn/rhn.conf` (affects patching).
- Increase the number of PostgreSQL connections available to Java applications (Tomcat, Taskomatic) according to the previous parameters, otherwise extra workers will starve waiting for a connection. Set parameter `hibernate.c3p0.max_size = 150` in `/etc/rhn/rhn.conf` (affects all minion operations). Make sure enough PostgreSQL connections are configured before changing this parameter - refer to `smdba system-check autotuning --help` to get automatic tuning of the PostgreSQL configuration file while changing the number of available connections. Additional manual tuning is usually not necessary but might be required depending on scale and exact use cases.
- Increase Salt's presence ping timeouts if responses might come back later than the defaults. Set parameters `salt_presence_ping_timeout = 20` and `salt_presence_ping_timeout = 20` in `/etc/rhn/rhn.conf` (affects all minion operations).
- Increase the Salt master worker threads allowing to parallelize more requests (otherwise Tomcat and Taskomatic workers will starve waiting for the Salt API). Set parameter `worker_threads: 100` in `/etc/salt/master.d/susemanager.conf` (affects onboarding and patching).
- Disable daily comparison of configuration files. Click on *Admin* › *Task Schedules*, then on the *compare-configs-default* link, then on the *Disable Schedule* button and finally on *Delete Schedule*.

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

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

6 Salt SSH Integration

Version 1, 2018-03-20

The following topic provides an overview of the Salt SSH (https://docs.saltstack.com/en/latest/topics/ssh/) integration with SUSE Manager. This integration adds support for both ssh-push and ssh-push-tunnel connections for Salt minions.

6.1 SSH Push Overview

Like the traditional stack, Salt minions may use an ssh connection to manage minions in place of Zeromq (https://docs.saltstack.com/en/latest/topics/transports/zeromq.html). This additional functionality is based on Salt SSH. Salt SSH enables you to execute salt commands and states via ssh without ever needing to install a salt-minion.

When the server executes an action on a minion an ssh connection is made on demand. This connection differs from the always-connected mode used by minions managed via Zeromq.

In SUSE Manager there are two ssh-push methods. In both use cases the server initiates an ssh connection to the minion in order to execute a Salt call using `salt-ssh`. The difference in the two methods is how `zypper/yum` initially connects to the server repositories:

ZYPPER CONNECTION METHODS:

ssh-push

zypper works as usual. The http(s) connection to the server is created directly.

ssh-push-tunnel

The server creates an http(s) connection through an ssh tunnel. The http(s) connection initiated by `zypper` is redirected through the tunnel by means of `/etc/hosts` aliasing (see below). This method should be used for in place firewall setups that block http(s) connections from a minion to the server.

6.2 Salt SSH Integration

As with all Salt calls, SUSE Manager invokes `salt-ssh` via the `salt-api`.

Salt SSH relies on a Roster to obtain details such as hostname, ports, and ssh parameters of an ssh minion. SUSE Manager keeps these details in the database and makes them available to Salt by generating a temporary Roster file for each salt-ssh call. The location of the temporary Roster file is supplied to salt-ssh using the `--roster-file= option`.

6.2.1 Authentication

salt-ssh supports both password and key authentication. SUSE Manager uses both methods:

PASSWORD AND KEY AUTHENTICATION:

Bootstrapping Authentication

Password authentication is used only when bootstrapping. During the bootstrap step the key of the server is not authorized on the minion and therefore a password must be utilized for a connection to be made. The password is used transiently in a temporary roster file used for bootstrapping. This password is not stored.

Common Salt Call Authentication

All other common salt calls use key authentication. During the bootstrap step the ssh key of the server is authorized on the minion (added to a minion's `~/.ssh/authorized_keys` file). Therefore subsequent calls no longer require a password.

6.2.2 User Account for salt-ssh Calls

The user for `salt-ssh` calls made by SUSE Manager is taken from the `ssh_push_sudo_user` setting. The default value of this is `root`.

If the value of `ssh_push_sudo_user` is not `root` then the `--sudo` options of `salt-ssh` are used.

6.3 SSH Push Tunnel HTTP(s) Redirection

For the `ssh-push-tunnel` method the traffic originating from zypper/yum has to be redirected through an ssh tunnel in order to bypass any firewall blocking a direct connection from the minion to the server.

This is achieved by using port `1233` in the repo url:

```
https://suma-server:1233/repourl...
```

Next alias the suma-server hostname to localhost in /etc/hosts:

```
127.0.0.1       localhost    suma-server
```

The server creates a reverse ssh tunnel that connects `localhost:1233` on the minion to `suma-server:443` (`ssh … -R 1233:suma-server:443`)

The result is that zypper/yum will actually connect to `localhost:1233` which is then forwarded to `suma-server:443` via the ssh tunnel.

This implies that zypper can contact the server only if the tunnel is open. This happens only when the servers executes an action on the minion. Manual zypper operations that require server connectivity are not possible in this case.

6.4 SUSE Manager Salt SSH Call Sequence

1. Prepare the Salt Roster for the call
   a. Create remote port forwarding option IF the contact method is ssh-push-tunnel
   b. Compute the ProxyCommand IF the minion is connected through a proxy
   c. create Roster content:
      - `hostname`
      - `user`
      - `port`
      - `remote_port_forwards`: The remote port forwarding ssh option
      - `ssh_options`: other ssh options:
        - `ProxyCommand`: If the minion connects through a SUMA proxy
      - `timeout`: default 180s
      - `minion_opts`:
        - `master`: set to the minion id if contact method is ssh-push-tunnel
2. create a temporary Roster file

3. execute a synchronous salt-ssh call via the API
4. remove the temporary Roster file

Additional Information:

SaltSSHService.callSyncSSH (https://github.com/SUSE/spacewalk/blob/Manager/java/code/src/com/suse/manager/webui/services/impl/SaltSSHService.java)

6.5 Bootstrap Process Sequence

Bootstrapping minions uses salt-ssh under the hood. This happens for both regular and ssh minion.

The bootstrap sequence is a bit different than the regular salt-ssh call:

1. For a regular minion generate and pre-authorize the Salt key of the minion
2. If this is an ssh minion and a proxy was selected retrieve the ssh public key of the proxy using the mgrutil.chain_ssh_cmd runner. The runner copies the public key of the proxy to the server using ssh. If needed it can chain multiple ssh commands to reach the proxy across multiple hops.
3. Generate pillar data for bootstrap. Pillar data contains:

   mgr_server
   : The hostname of the SUSE Manager server

   minion_id
   : The hostname of the minion to bootstrap

   contact_method
   : The connection type

   mgr_sudo_user
   : The user for salt-ssh

   activation_key
   : If selected

   minion_pub
   : The public minion key that was pre-authorized

minion_pem
: The private minion key that was pre-authorized

proxy_pub_key
: The public ssh key that was retrieved from the proxy if the target is an ssh minion and a proxy was selected

4. If contact method is `ssh-push-tunnel` fill the remote port forwarding option
5. if the minion connects through a SUMA proxy compute the `ProxyCommand` option. This depends on the path used to connect to the proxy, e.g. server → proxy1 → proxy2 → minion
6. generate the roster for bootstrap into a temporary file. This contains:
   - `hostname`
   - `user`
   - `password`
   - `port`
   - `remote_port_forwards`: the remote port forwarding ssh option
   - `ssh_options`: other ssh options:
     - `ProxyCommand` if the minion connects through a SUMA proxy
   - `timeout`: default 180s
7. Via the Salt API execute:

```
salt-ssh --roster-file=<temporary_bootstrap_roster> minion state.apply
 certs,<bootstrap_state>`
```

Note

<bootstrap_state> replaceable by **bootstrap** for regular minions or **ssh_bootstrap** for ssh minions.

The following image provides an overview of the Salt SSH bootstrap process.

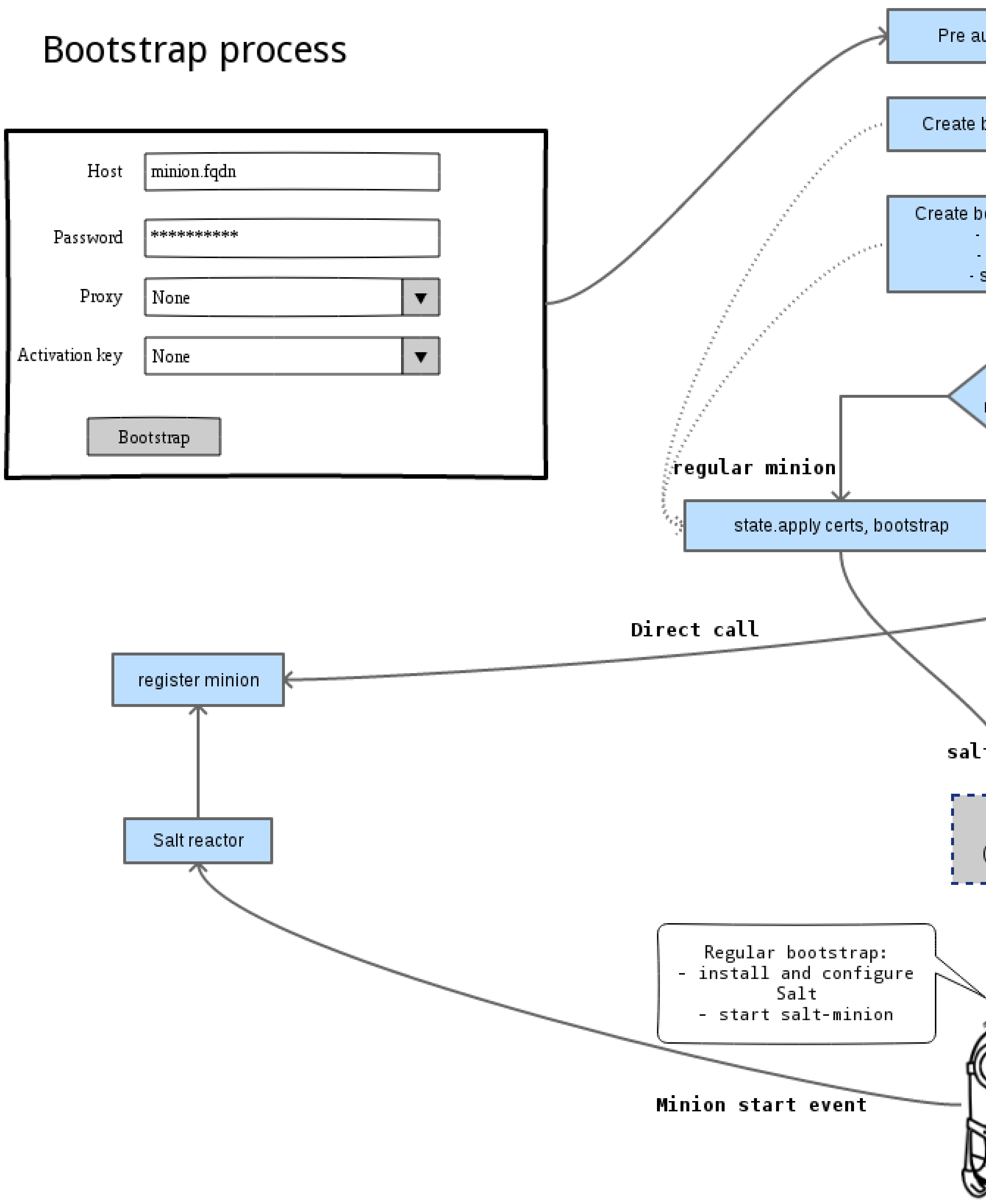


FIGURE 6.1: SALT SSH BOOTSTRAP PROCESS

Additional Information:

- SSHMinionBootstrapper.java (https://github.com/SUSE/spacewalk/blob/Manager/java/code/src/com/suse/manager/webui/controllers/utils/RegularMinionBootstrapper.java) ↗
- RegularMinionBootstrapper.java (https://github.com/SUSE/spacewalk/blob/Manager/java/code/src/com/suse/manager/webui/controllers/utils/SSHMinionBootstrapper.java) ↗
- bootstrap/init.sls (https://github.com/SUSE/spacewalk/blob/Manager/susemanager-utils/susemanager-sls/salt/bootstrap/init.sls) ↗
- ssh_bootstrap/init.sls (https://github.com/SUSE/spacewalk/blob/Manager/susemanager-utils/susemanager-sls/salt/ssh_bootstrap/init.sls) ↗

6.6 Proxy Support

In order to make salt-ssh work with SUSE Managers proxies the ssh connection is chained from one server/proxy to the next. This is also know as multi-hop or multi gateway ssh connection.

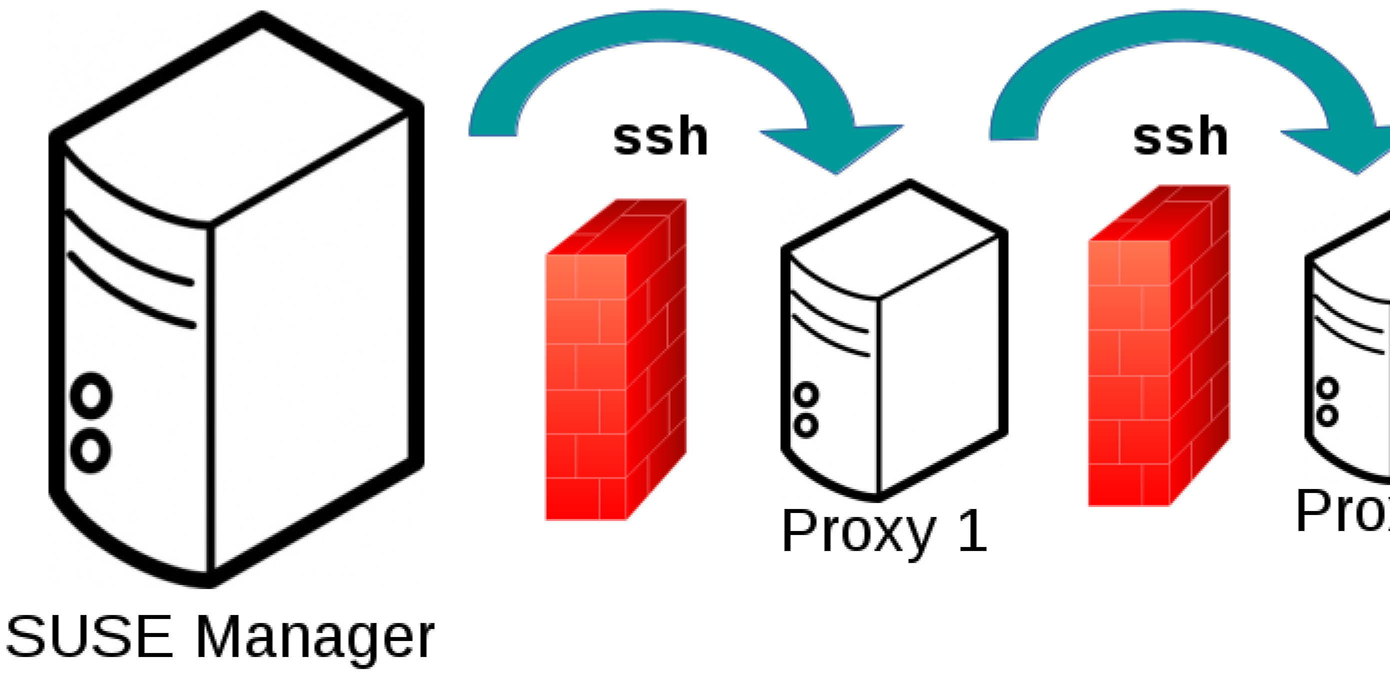


FIGURE 6.2: SALT SSH PROXY MULTIPLE HOPS

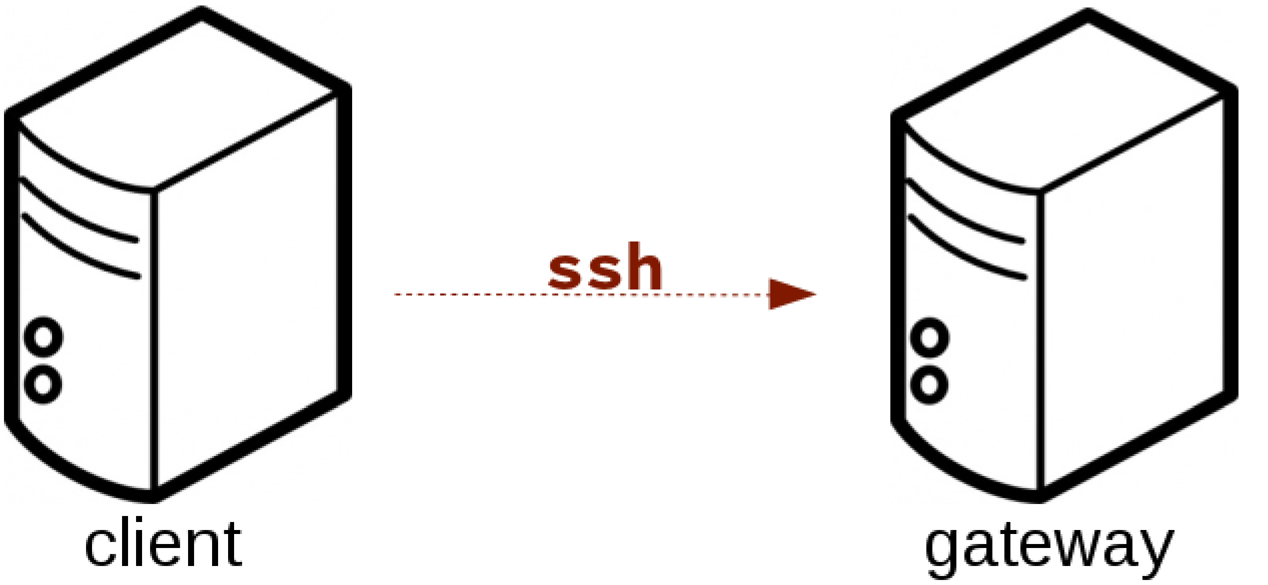
6.6.1 The ProxyCommand

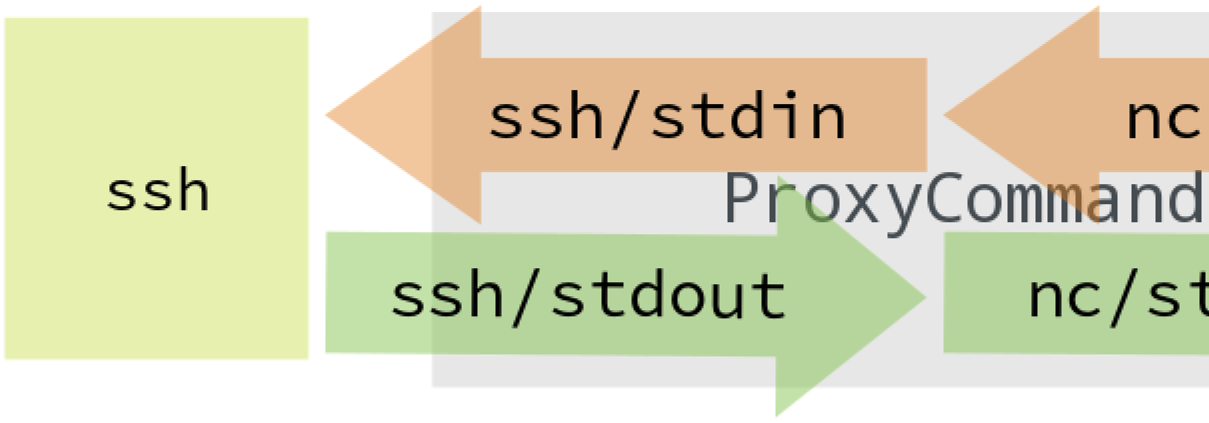
In order to redirect the ssh connection through the proxies the ssh `ProxyCommand` option is used. This options invokes an arbitrary command that is expected to connect to the ssh port on the target host. The standard input and output of the command is used by the invoking ssh process to talk to the remote ssh daemon.

The ProxyCommand basically replaces the TCP/IP connection. It doesn't do any authorization, encryption, etc. Its role is simply to create a byte stream to the remote ssh daemon's port.

E.g. connecting to a server behind a gateway:

```
ssh -o ProxyCommand=<stdio/std
ssh -o ProxyCommand='ssh gateway nc f
```





Note

In this example netcat (nc) is used to pipe port 22 of the target host into the ssh std i/o.

6.6.2 Salt SSH Call Sequence via Proxy

1. SUSE Manager initates the ssh connections as described above.
2. Additionally the ProxyCommand uses ssh to create a connection from the server to the minion through the proxies.

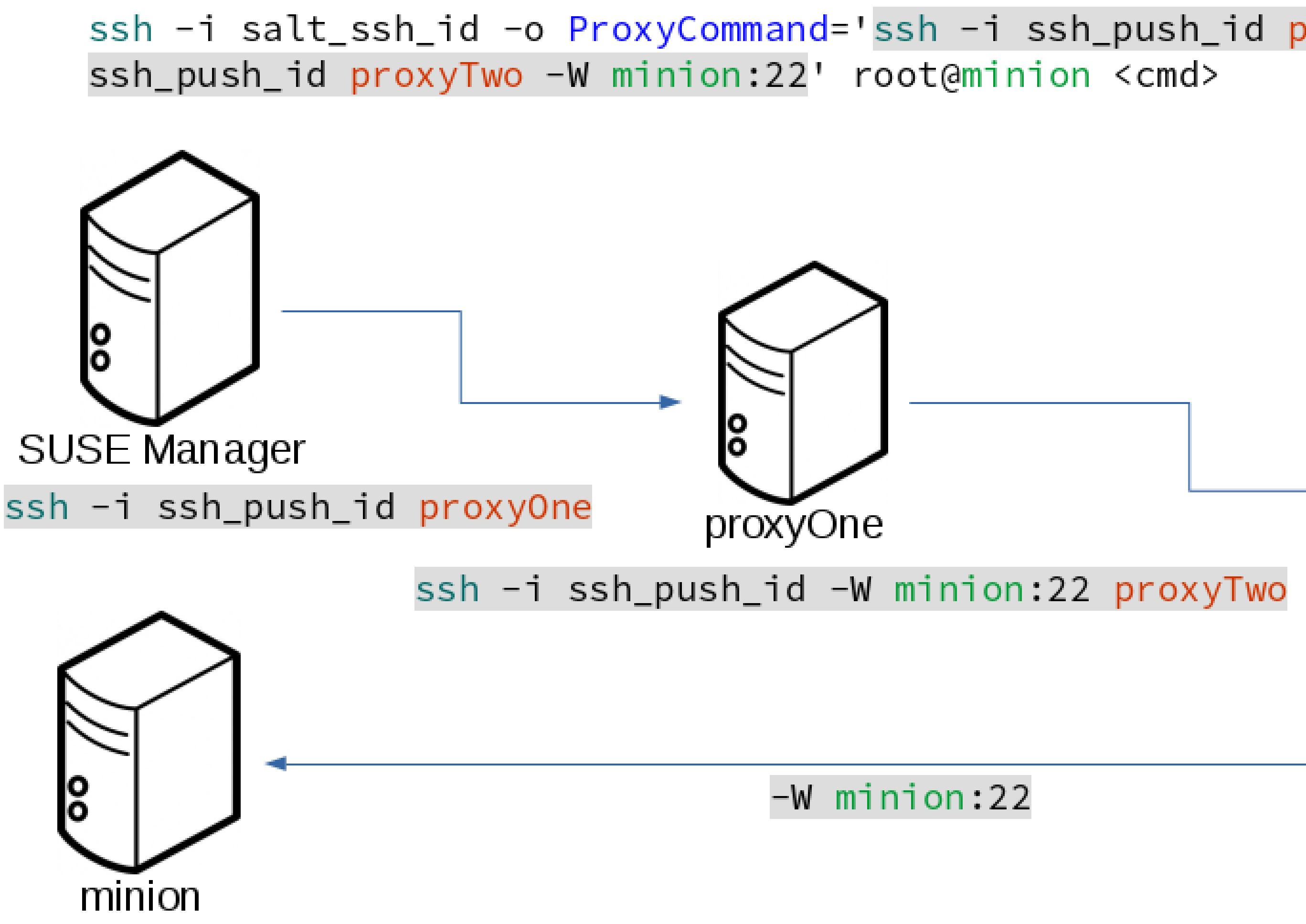
6.6.2.1 Twin Proxies and SSH Push

The following example uses the ProxyCommand option with two proxies and the usual ssh-push method:

```
# 1
/usr/bin/ssh -i /srv/susemanager/salt/salt_ssh/mgr_ssh_id -o StrictHostKeyChecking=no -o
 User=mgrsshtunnel  proxy1
# 2
/usr/bin/ssh -i /var/lib/spacewalk/mgrsshtunnel/.ssh/id_susemanager_ssh_push -o
 StrictHostKeyChecking=no -o User=mgrsshtunnel -W minion:22  proxy2
```

STEPS

1. connect from the server to the first proxy
2. connect from the first proxy to the second and forward standard input/output on the client to minion:22 using the -W option.



```
ssh -i salt_ssh_id -o ProxyCommand='ssh -i ssh_push_id p
ssh_push_id proxyTwo -W minion:22' root@minion <cmd>
```

```
ssh -i ssh_push_id proxyOne
```

```
ssh -i ssh_push_id -W minion:22 proxyTwo
```

```
-W minion:22
```

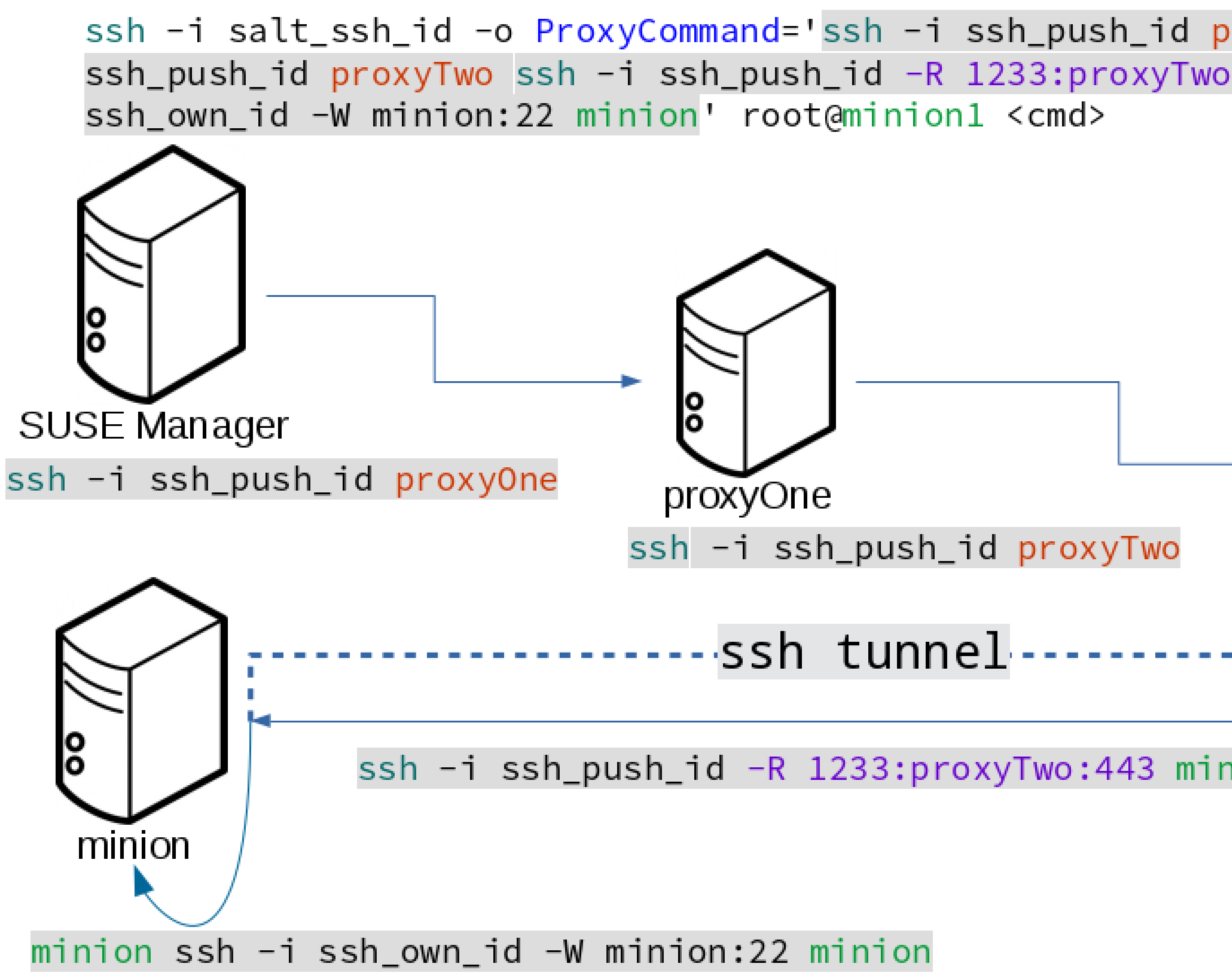
6.6.2.2 Twin Proxies and SSH Push Tunnel

The following example uses the ProxyCommand option with two proxies over an ssh-push-tunnel connection:

```
# 1
/usr/bin/ssh -i /srv/susemanager/salt/salt_ssh/mgr_ssh_id -o User=mgrsshtunnel  proxy1
# 2
/usr/bin/ssh -i /home/mgrsshtunnel/.ssh/id_susemanager_ssh_push -o User=mgrsshtunnel
 proxy2
# 3
/usr/bin/ssh -i /home/mgrsshtunnel/.ssh/id_susemanager_ssh_push -o User=root -R
 1233:proxy2:443 minion
# 4
/usr/bin/ssh -i /root/.ssh/mgr_own_id -W minion:22 -o User=root minion
```

STEPS

1. Connect from the server to the first proxy.
2. Connect from the first proxy to the second.
3. connect from the second proxy to the minion and open an reverse tunnel (-R 1233:proxy2:443) from the minion to the https port on the second proxy.
4. Connect from the minion to itself and forward the std i/o of the server to the ssh port of the minion (-W minion:22). This is equivalent to ssh … proxy2 netcat minion 22 and is needed because ssh doesn't allow to have both the reverse tunnel (-R 1233:proxy2:443) and the standard i/o forwarding (-W minion:22) in the same command.


ssh -i salt_ssh_id -o ProxyCommand='ssh -i ssh_push_id p
ssh_push_id proxyTwo ssh -i ssh_push_id -R 1233:proxyTwo
ssh_own_id -W minion:22 minion' root@minion1 <cmd>
SUSE Manager
ssh -i ssh_push_id proxyOne
proxyOne
ssh -i ssh_push_id proxyTwo
ssh tunnel
ssh -i ssh_push_id -R 1233:proxyTwo:443 min
minion
minion ssh -i ssh_own_id -W minion:22 minion

Additional Information:

- SaltSSHService.sshProxyCommandOption (https://github.com/SUSE/spacewalk/blob/Manager/java/code/src/com/suse/manager/webui/services/impl/SaltSSHService.java) ↗

6.7 Users and SSH Key Management

In order to connect to a proxy the parent server/proxy uses a specific user called `mgrsshtunnel`.

The ssh config `/etc/ssh/sshd_config` of the proxy will force the execution of `` `/usr/sbin/mgr-proxy-ssh-force-cmd `` when `mgrsshtunnel` connects.

`` `/usr/sbin/mgr-proxy-ssh-force-cmd `` is a simple shell script that allows only the execution of `scp`, `ssh` or `cat` commands.

The connection to the proxy or minion is authorized using ssh keys in the following way:

1. The server connects to the minion and to the first proxy using the key in `` `/srv/susemanager/salt/salt_ssh/mgr_ssh_id ``.
2. Each proxy has its own key pair in `` `/home/mgrsshtunnel/.ssh/id_susemanager_ssh_push ``.
3. Each proxy authorizes the key of the parent proxy or server.
4. The minion authorized its own key.

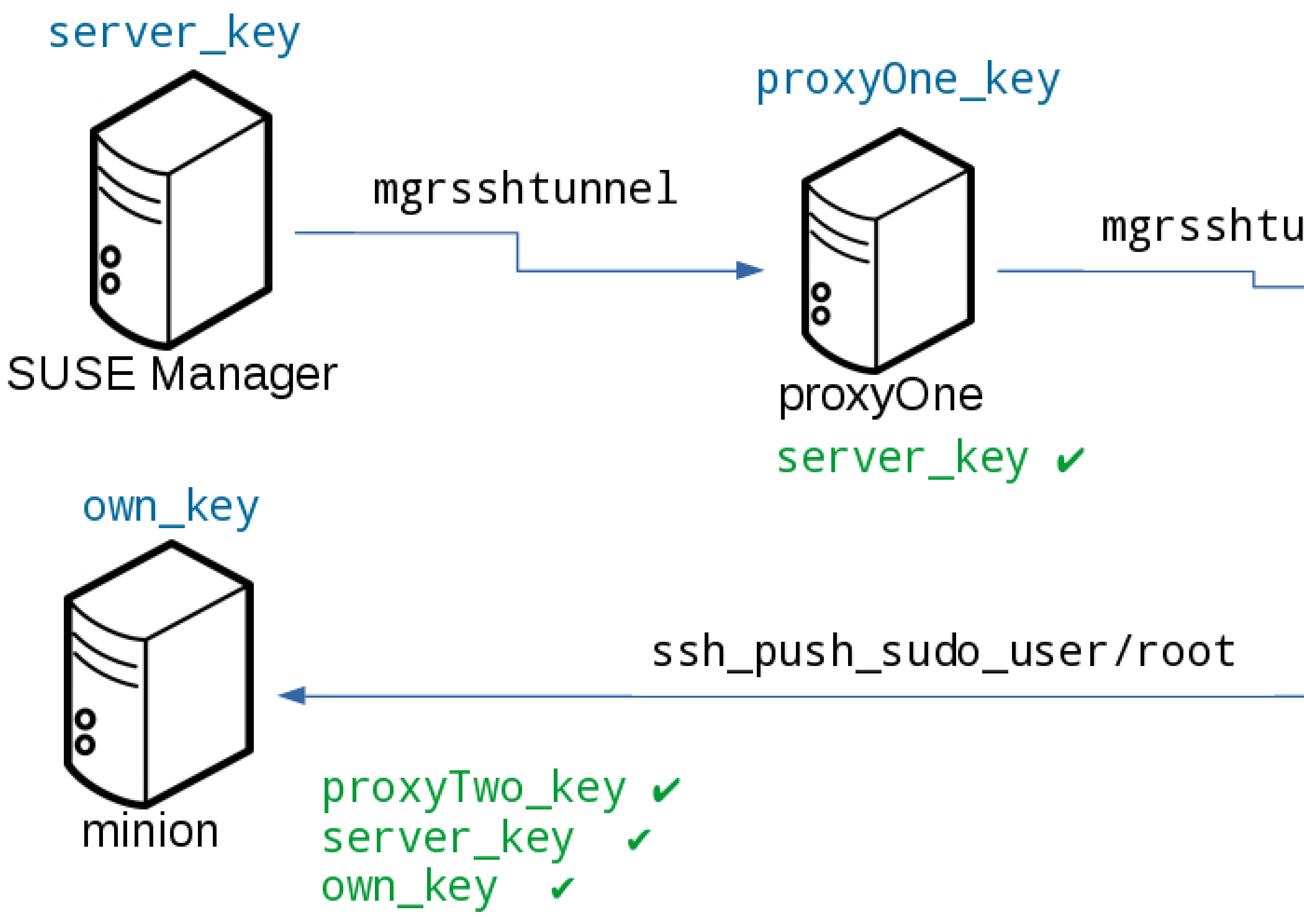


FIGURE 6.3: SALT SSH KEY AUTHORIZATION PROCESS

Additional Information:

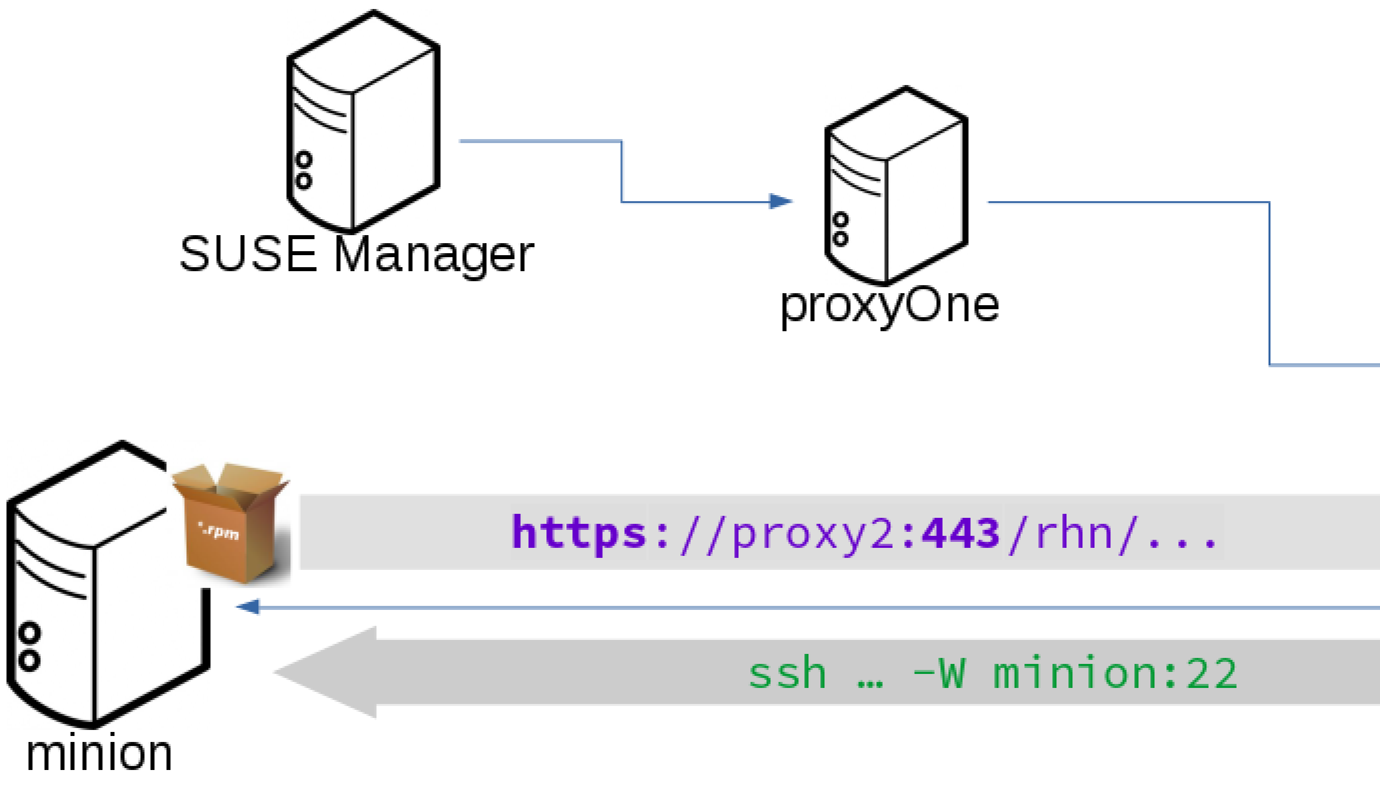
- mgr-proxy-ssh-force-cmd (https://github.com/SUSE/spacewalk/blob/Manager/proxy/proxy/mgr-proxy-ssh-force-cmd) ↗

6.8 Repository access via proxy

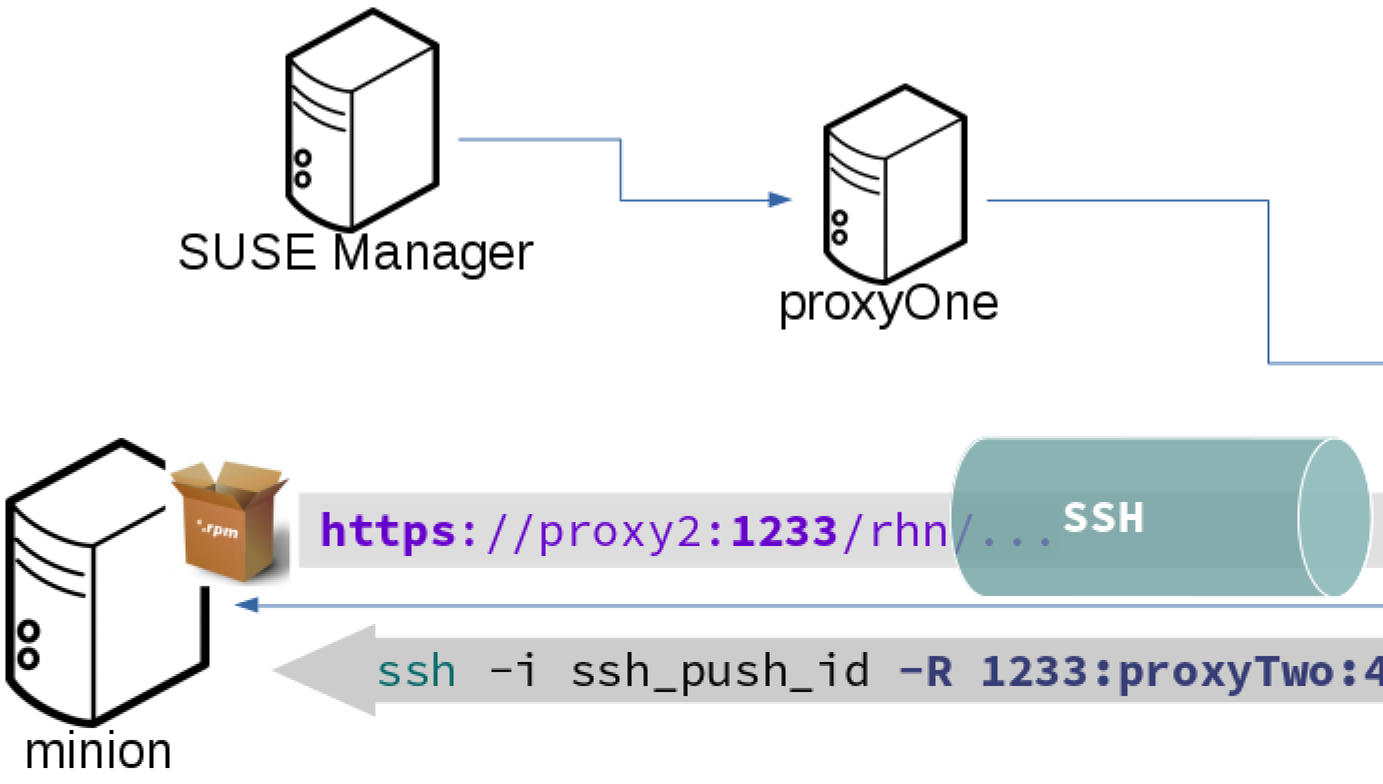
For both ssh-push and ssh-push-tunnel the minion connects to the proxy to retrieve packages and repo data.

The difference is how the connection works:

- In case of ssh-push, zypper or yum connect directly to the proxy using http(s). This assumes there's not firewall between the minion and the proxy that would block http connections initiated by the minion.



- In case of ssh-push-tunnel, the http connection to the proxy is redirected through a reverse ssh tunnel.


SUSE Manager
proxyOne
minion
*.rpm
https://proxy2:1233/rhn/...
SSH
ssh -i ssh_push_id -R 1233:proxyTwo:4

6.9 Proxy setup

When the `spacewalk-proxy` package is installed on the proxy the user `mgrsshtunnel` is created if it doesn't already exist.

During the initial configuration with `configure-proxy.sh` the following happens:

1. Generate a ssh key pair or import an existing one
2. Retrieve the ssh key of the parent server/proxy in order to authorize it on the proxy
3. Configure the `sshd` of the proxy to restrict the user `mgrsshtunnel`

This configuration is done by the `mgr-proxy-ssh-push-init` script. This is called from `configure-proxy.sh` and the user doesn't have to invoke it manually.

Retrieving the parent key is done by calling an HTTP endpoint on the parent server or proxy.

1. First `https//$PARENT/pub/id_susemanager_ssh_push.pub` is tried. If the parent is proxy this will return the public ssh key of that proxy.
2. If a `404` is received then it's assumed the parent is a server not a proxy and `https://$PARENT/rhn/manager/download/saltssh/pubkey` is tried.
   a. If `/srv/susemanager/salt/salt_ssh/mgr_ssh_id.pub` already exists on the server it's returned
   b. If the public key doesn't exist (because `salt-ssh` has not been invoked yet) generate the key by calling the `mgrutil.ssh_keygen` runner

Note

salt-ssh generates a key pair the first time it is invoked in `/srv/susemanager/salt/salt_ssh/mgr_ssh_id`. The previous sequence is needed in case a proxy is configured before salt-ssh was invoked for the first time.

Additional Information:

- com.suse.manager.webui.controllers.SaltSSHController (https://github.com/SUSE/spacewalk/blob/Manager/java/code/src/com/suse/manager/webui/controllers/SaltSSHController.java)
- mgrutil.ssh_keygen (https://github.com/SUSE/spacewalk/blob/Manager/susemanager-utils/susemanager-sls/modules/runners/mgrutil.py)
- mgr-proxy-ssh-push-init (https://github.com/SUSE/spacewalk/blob/Manager/proxy/proxy/mgr-proxy-ssh-push-init)
- spacewalk-proxy.spec (https://github.com/SUSE/spacewalk/blob/Manager/proxy/proxy/spacewalk-proxy.spec)

7 Monitoring with Icinga

7.1 Introduction

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

7.2 Installation and Basic Configuration

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

Tip: Icinga Installation Location

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

PROCEDURE: INSTALLATION AND BASIC CONFIGURATION

1. Register the new client with Uyuni and subscribe it to the Uyuni client and update channels. SLES 12 and later include these channels by default.

2. Install the required Icinga packages on the new client:

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

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

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

4. Enable postgres on boot and start the database:

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

5. Create the database and user for Icinga:

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

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

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

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

Important: Placement of Authentication Settings

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

7. Reload the Postgres service:

```
systemctl reload postgresql.service
```

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

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

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

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

 db_servertype=pgsql
 db_port=5432
```

Important: Open Firewall Port

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

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

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

11. Enable and start all required services:

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

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

This concludes setup and initial configuration of Icinga.

7.3 Icinga and NRPE Quickstart

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

PROCEDURE: ADDING UYUNITO ICINGA FOR MONITORING

1. On your Uyuni server install the required packages:

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

2. Modify the NRPE configuration file located at:

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

   Edit or add the following lines:

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

   Variable definitions:

   server_port

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

   nrpe_user

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

   allowed_hosts

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

dont_blame_nrpe

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

command[check_systemd.sh]

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

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

```
systemctl start nrpe
```

This concludes setup of nrpe.

7.3.1 Add a Host to Icinga

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

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

Note

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

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

```
systemctl restart icinga
```

7.3.2 Adding Services to Icinga

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

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

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

```
systemctl restart icinga
```

7.3.3 Creating Icinga Hostgroups

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

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

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

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

```
systemctl restart icinga
```

7.3.4 Creating Icinga Servicegroups

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

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

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

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

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

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

```
systemctl restart icinga
```

7.4 Monitoring Systemd Services

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

PROCEDURE: MONITORING RUNNING SYSTEMD SERVICES

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

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

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

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

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


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

   service=$1

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

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

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

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

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

Warning: Non-supported 3rd Party Plugin

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

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

3. Make the script executable:

```
chmod 755 check_systemd.sh
```

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

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

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

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

```
visudo
```

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

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

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

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

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

7. Now you will add the following critical services to be montitored to your Uyuni host file:

   - auditlog-keeper.service
   - jabberd.service
   - spacewalk-wait-for-jabberd.service
   - tomcat.service
   - spacewalk-wait-for-tomcat.service
   - salt-master.service
   - salt-api.service
   - spacewalk-wait-for-salt.service
   - apache2.service
   - osa-dispatcher.service
   - rhn-search.service
   - cobblerd.service
   - taskomatic.service
   - spacewalk-wait-for-taskomatic.service

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

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

}

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

}

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

# Monitor Tomcat
define service{
       use                     generic-service
       host_name               susemanager
       check_interval          1
       active_checks_enabled   1
```

```
        service_description    Tomcat Service
        servicegroups          SUSE Manager Essential Services
        check_command          check-systemd-service!tomcat.service
}

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

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

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

# Monitor Spacewalk Wait for Salt
define service{
        use                    generic-service
        host_name              susemanager
        check_interval         1
        active_checks_enabled  1
        service_description    Spacewalk Wait For Salt Service
        servicegroups          SUSE Manager Essential Services
```

```
       check_command          check-systemd-service!spacewalk-wait-for-
salt.service
}

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

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

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

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

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

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

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

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

8. Restart Icinga:

```
systemctl restart icinga
```

7.5 Using the check_suma_patches Plugin

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

PROCEDURE: SETUP CHECK_SUMA_PATCHES

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

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

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

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

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

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

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

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

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

This concludes setup of the `check_suma_patches` plugin.

7.6 Using the check_suma_lastevent Plugin

You can use the `check_suma_lastevent` plugin to display the last action executed on any host. The following procedure will guide you through the setup of the check_suma_patches plugin.

PROCEDURE: SETUP CHECK_SUMA_LASTEVENT

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

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

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

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

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

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

4. Status will be reported as follows:

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

This concludes setup of the `check_suma_lastevent` plugin.

7.7 Additional Resources

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

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

8 Image Building and Management

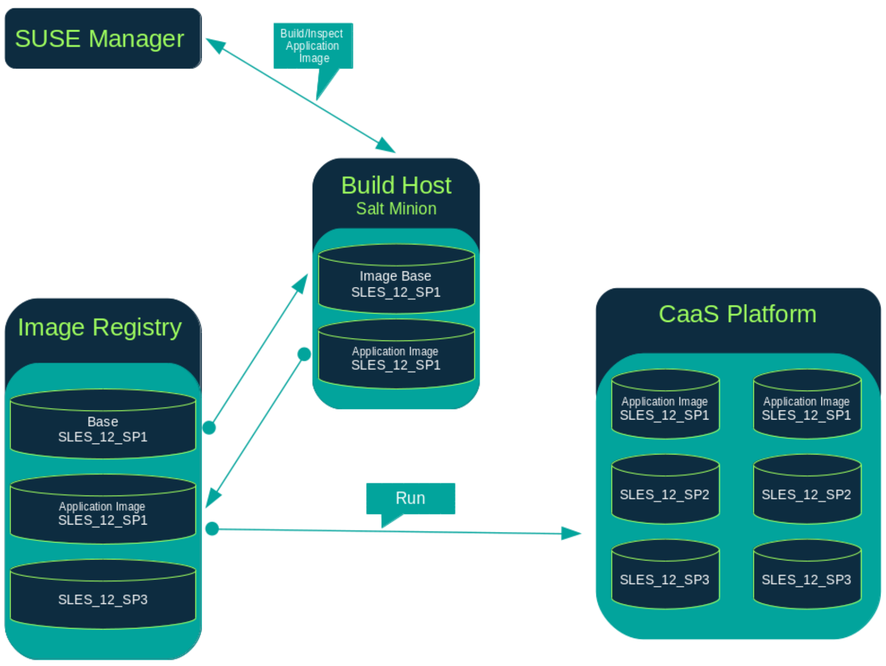
8.1 Image Building Overview

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

Uyuni supports two distinct build types:

- Dockerfile-for more information, see *Section 8.2, "Container Images"*
- Kiwi image system-for more information, see *Section 8.3, "OS Images"*

8.2 Container Images



8.2.1 Requirements

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

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

> **Note: Registry Provider Solutions**
>
> If you require a private image registry you can use an open source solution such as `Portus`. For additional information on setting up Portus as a registry provider, see the Portus Documentation (http://port.us.org/).

For more information on Containers or CaaS Platform, see:

- SUSE Linux Enterprise Server 12 SP3 Docker Guide (https://www.suse.com/documentation/sles-12/book_sles_docker/data/book_sles_docker.html)
- SUSE CaaS Platform 2 Documentation (https://www.suse.com/documentation/suse-caasp-2/)

8.2.2 Creating a Build Host

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

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

1. Select a minion to be designated as a build host from the *Systems* › *Overview* page.
2. From the `System Details` page for the selected minion assign the containers modules by going to *Software* › *Software Channels* and enabling `SLE-Module-Containers12-Pool` and `SLE-Module-Containers12-Updates`. Confirm by clicking *Change Subscriptions*.

3. From the *System Details › Properties* page, enable `Add-on System Type` and `Container Build Host` and confirm by clicking *Update Properties*.

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

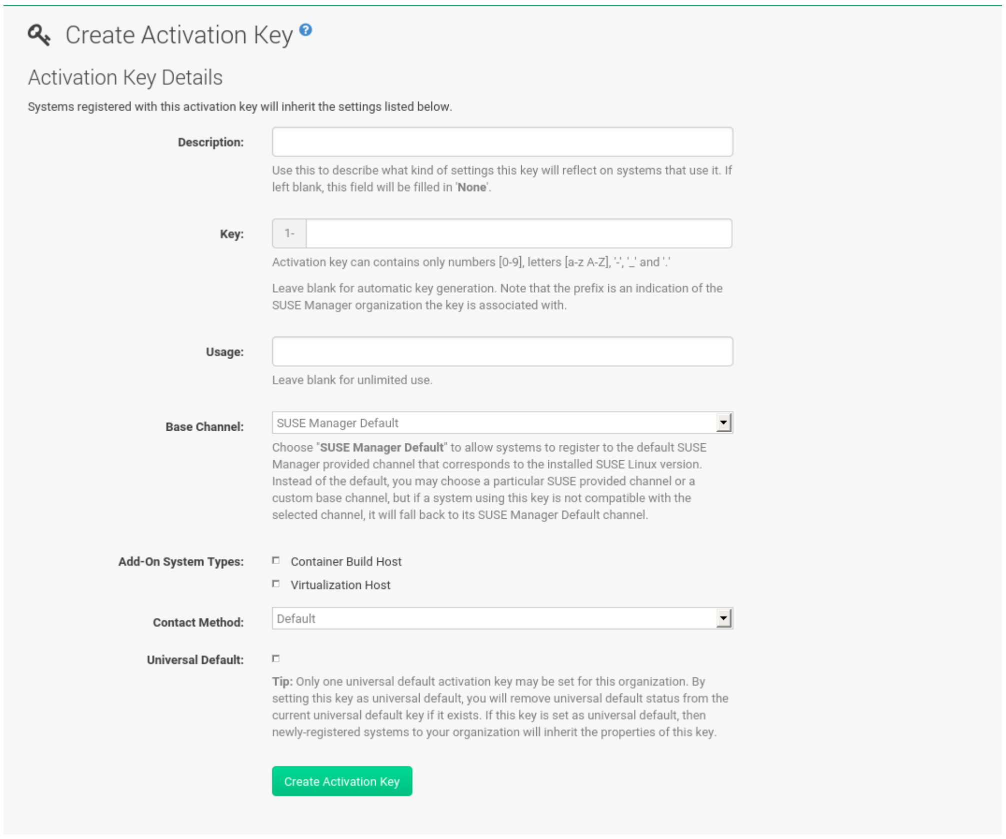
```
salt '$your_minion' state.highstate
```

8.2.2.1 Define Container Build Channels with an Activation Key

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

Note: Relationship Between Activation Keys and Image Profiles

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


Create Activation Key

Activation Key Details

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

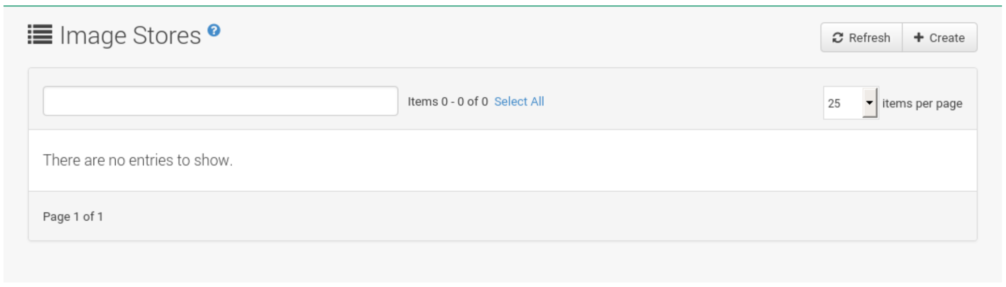
Description:	Use this to describe what kind of settings this key will reflect on systems that use it. If left blank, this field will be filled in '**None**'.
Key:	1- Activation key can contains only numbers [0-9], letters [a-z A-Z], '-', '_' and '.' Leave blank for automatic key generation. Note that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

1. Select *Main Menu* › *Systems* › *Activation Keys*.
2. Click *Create Key*.
3. Enter a `Description` and a `Key` name. Use the drop-down menu to select the `Base Channel` to associate with this key.
4. Confirm with *Create Activation Key*.

For more information, see *Book "Best Practices", Chapter 7 "Activation Key Management"*.

8.2.3 Creating an Image Store

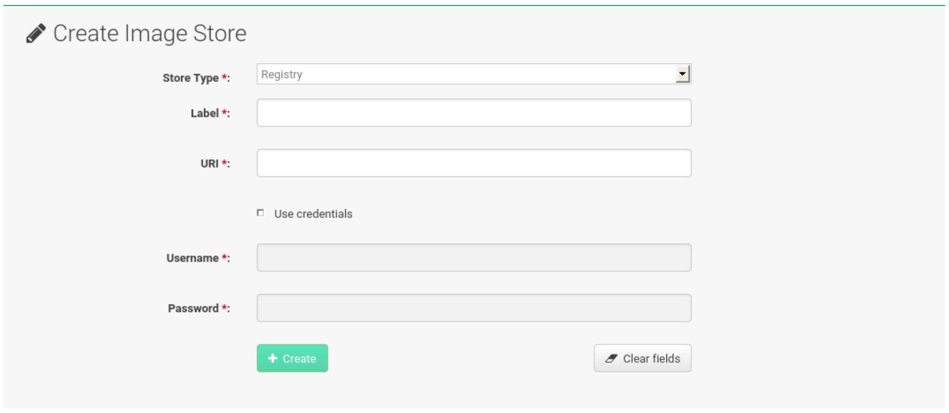
Define a location to store all of your images by creating an Image Store.



There are no entries to show.

Page 1 of 1

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



Store Type *: Registry

Label *:

URI *:

Use credentials

Username *:

Password *:

3. Uyuni currently provides support only for the `Registry` store type. Define a name for the image store in the `Label` field.

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

```
registry.example.com
```

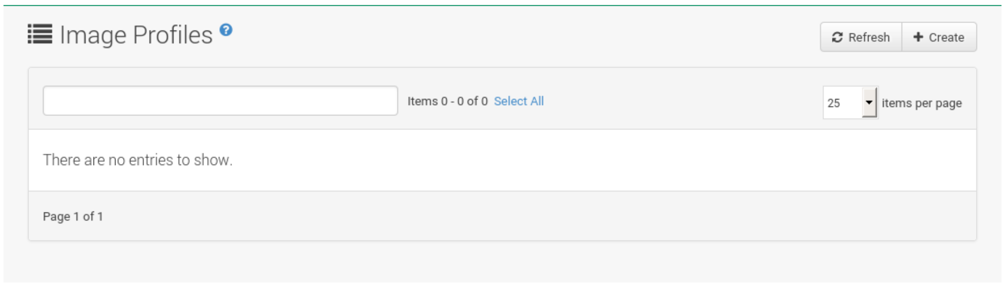
   The Registry URI can also be used to specify an image store on a used registry.

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

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

8.2.4 Creating an Image Profile

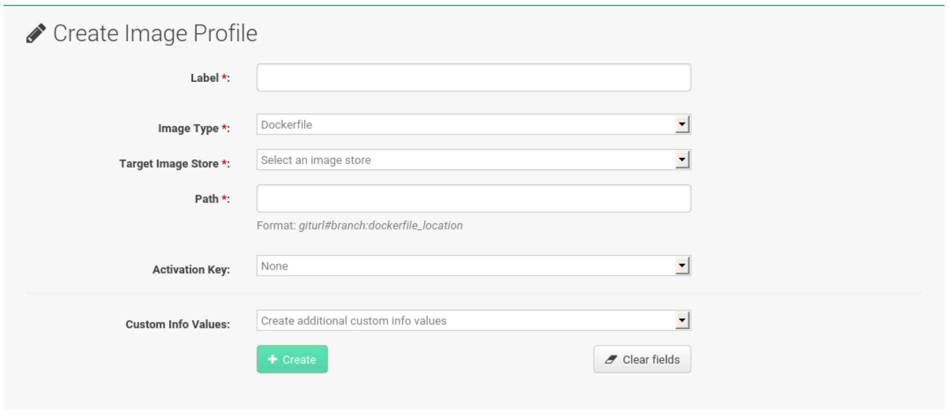
Manage Image Profiles from the `Image Profile` page.



There are no entries to show.

PROCEDURE: CREATE AN IMAGE PROFILE

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



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

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

Note

Only lowercase characters are permitted in container labels. If your container image tag is in a format such as `myproject/myimage`, make sure your image store registry URI contains the `/myproject` suffix.

3. Use a `Dockerfile` as the `Image Type`
4. Use the drop-down menu to select your registry from the `Target Image Store` field.
5. Enter a Github or Gitlab repository URL (http, https, or token authentication) in the `Path` field using one of the following formats:

GITHUB PATH OPTIONS

- Github single user project repository

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

- Github organization project repository

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

- Github token authentication:

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

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

GITLAB PATH OPTIONS

- Gitlab single user project repository

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

- Gitlab groups project repository

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

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

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

Important: Specifying a Github or Gitlab Branch

If a branch is not specified, the `master` branch will be used by default. If a `folder` is not specified the image sources (`Dockerfile` sources) are expected to be in the root directory of the Github or Gitlab checkout.

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

   #### Note: Relationship Between Activation Keys and Image Profiles

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

2. Click the *Create* button.

8.2.5 Example Dockerfile and add_packages Script

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

Important: Required Dockerfile Lines

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

For example: The `ARG repo` parameter and the `echo` command pointing to the repository file, creates and then injects the correct path into the repository file for the desired channel version.

The repository version is determined by the activation key that you assigned to your image profile.

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

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

ARG repo
ARG cert

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

# Update certificate trust store
RUN update-ca-certificates

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

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

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

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

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

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

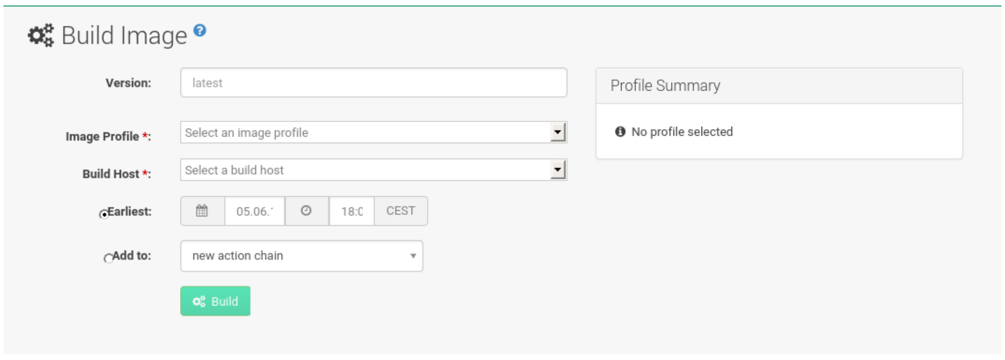
```
#!/bin/bash
set -e

zypper --non-interactive --gpg-auto-import-keys ref
zypper --non-interactive in python python-xml aaa_base aaa_base-extras net-tools timezone
 vim less sudo tar
```

Note: Packages Required for Inspecting Your Images

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

8.2.6 Building an Image

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


Build Image

Version:	latest
Image Profile *:	Select an image profile
Build Host *:	Select a build host
Earliest:	05.06.· 18:0 CEST
Add to:	new action chain

Profile Summary

No profile selected

PROCEDURE: BUILD AN IMAGE

1. For this example select *Images* › *Build*.
2. Add a different tag name if you want a version other than the default `latest` (only relevant to containers).
3. Select `Build Profile` and `Build Host`.

Note: Profile Summary

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

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

8.2.7 Importing an Image

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

PROCEDURE: IMPORT AN IMAGE

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

   Image store
   The registry from where the image will be pulled for inspection.

   Image name
   The name of the image in the registry.

   Image version
   The version of the image in the registry.

   Build host
   The build host that will pull and inspect the image.

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

   For confirmation, click *Import*.

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

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

8.2.8 Troubleshooting

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

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

8.3 OS Images

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

For more information about the Kiwi build system, see the Kiwi documentation (https://doc.opensuse.org/projects/kiwi/doc/)↗.

8.3.1 Enabling OS Image Building

OS image building is disabled by default. You can enable it in `/etc/rhn/rhn.conf` by setting:

```
java.kiwi_os_image_building_enabled = true
```

Restart the Spacewalk service to pick up the changes:

```
spacewalk-service restart
```

8.3.2 Requirements

The Kiwi image building feature is available for Salt minions running SUSE Linux Enterprise Server 12 or later.

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

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

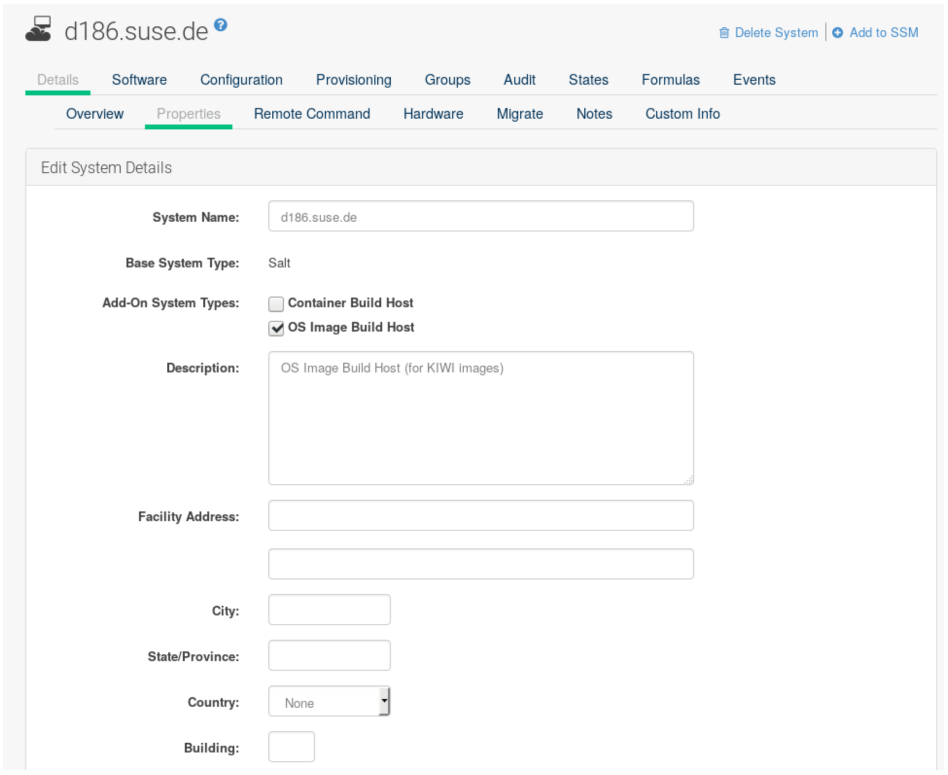
Example scripts are provided in the following sections.

8.3.3 Creating a Build Host

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

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

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



Edit System Details

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

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

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

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

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

When you upgrade the `spacewalk-certs-tools` package, the upgrade scenario will call the package script using the default values. However if the certificate path was changed or unavailable, you will need to call the package script manually using `--ca-cert-full-path <path_to_certificate>` after the upgrade procedure has finished.

Package script call example.

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

```
ORG-TRUSTED-SSL-CERT
```

The RPM package with the certificate is stored in a salt-accessible directory such as `/usr/share/susemanager/salt/images/rhn-org-trusted-ssl-cert-osimage-1.0-1.noarch.rpm`.

The RPM package with the certificate is provided in the local build host repository `/var/lib/Kiwi/repo`.

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

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

config.xml.

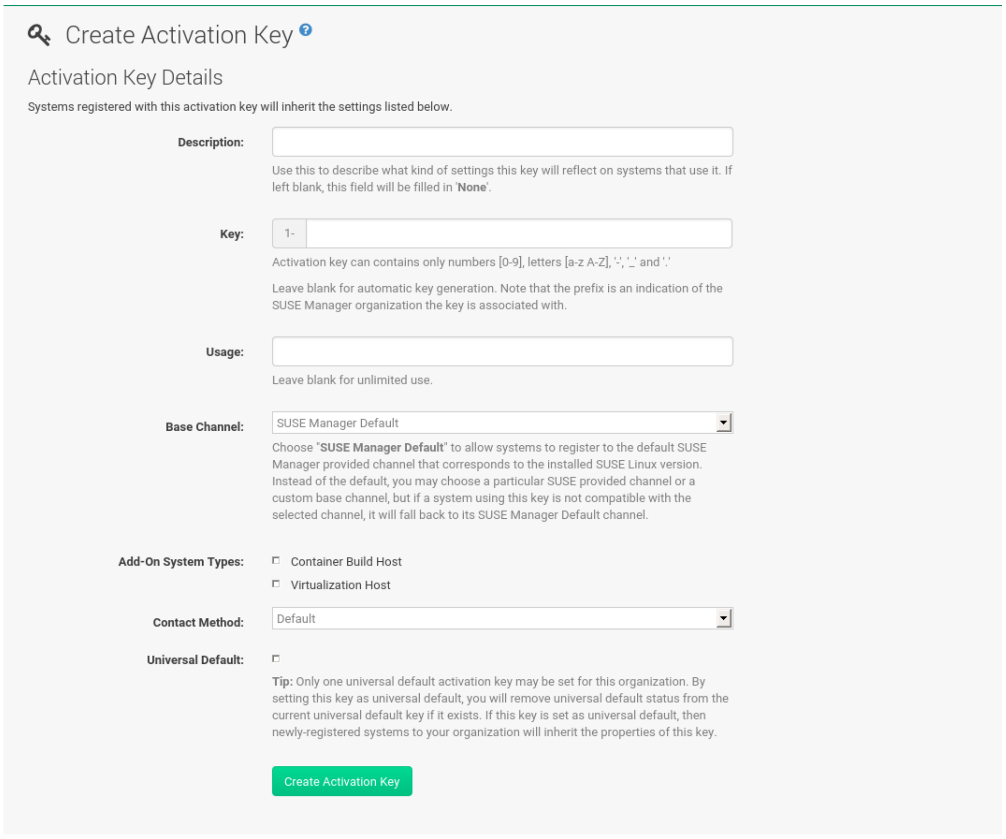
```
...
  <packages type="bootstrap">
    ...
    <package name="rhn-org-trusted-ssl-cert-osimage" bootinclude="true"/>
  </packages>
...
```

8.3.3.1 Define Kiwi Build Channels with an Activation Key

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

Note: Relationship Between Activation Keys and Image Profiles

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


Create Activation Key

Activation Key Details

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

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

1. In the Web UI, select *Main Menu › Systems › Activation Keys*.
2. Click `Create Key`.
3. Enter a `Description`, a `Key` name, and use the drop-down box to select a `Base Channel` to associate with the key.
4. Confirm with *Create Activation Key*.

For more information, see *Book “Best Practices”, Chapter 7 “Activation Key Management”*.

8.3.4 Image Store

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

Note: Image stores for Kiwi build type

Image stores for Kiwi build type, used to build system, virtual and other images, are not supported yet.

Images are always stored in `/srv/www/os-images/<organization id>` and are accessible via HTTP/HTTPS `https://<susemanager_host>/os-images/<organization id>`

8.3.5 Creating an Image Profile

Manage Image Profiles using the Web UI.

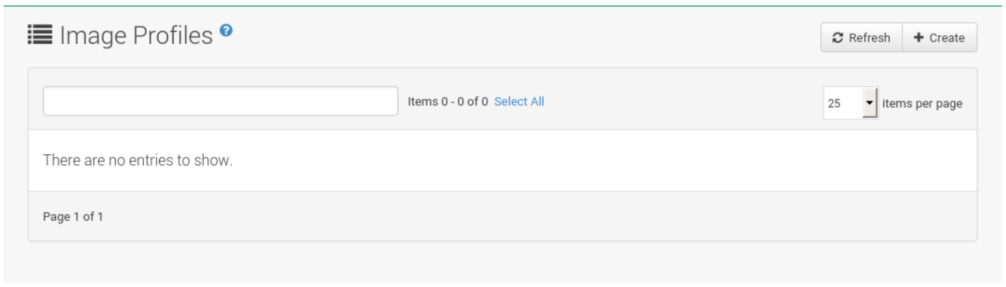


Image Profiles

Items 0 - 0 of 0 Select All

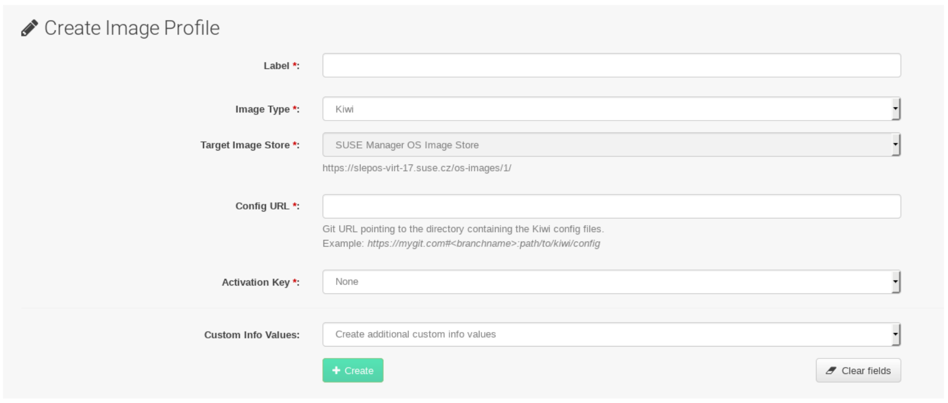
25 items per page

There are no entries to show.

Page 1 of 1

PROCEDURE: CREATE AN IMAGE PROFILE

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



Create Image Profile

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

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

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

4. Image store is automatically selected.

5. Enter a `Config URL` to the directory containing the Kiwi configuration files:
    a. Git URI
    b. HTTPS tarball
    c. Path to build host local directory
6. Select an `Activation Key`. Activation keys ensure that images using a profile are assigned to the correct channel and packages.

    Note: Relationship Between Activation Keys and Image Profiles

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

7. Confirm with the *Create* button.

SOURCE FORMAT OPTIONS

- Git/HTTP(S) URL to the repository
  URL to the Git repository containing the sources of the image to be built. Depending on the layout of the repository the URL can be:

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

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

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

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

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

- HTTP(S) URL to the tarball
  URL to the tar archive, compressed or uncompressed, hosted on the webserver.

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

- Path to the directory on the build host

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

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

8.3.6 Example of Kiwi sources

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

For information about the Kiwi build system, see the Kiwi documentation (https://doc.opensuse.org/projects/kiwi/doc/).

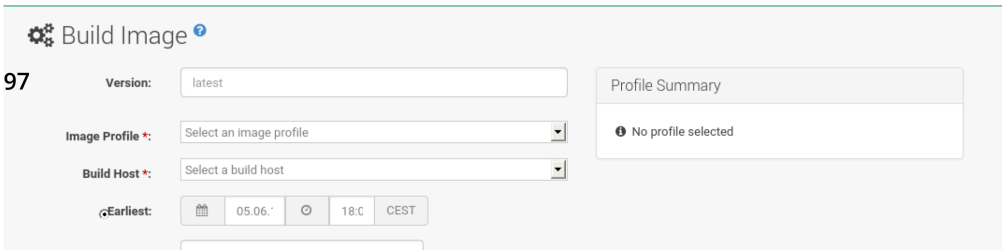
SUSE provides examples of fully functional image sources at the SUSE/manager-build-profiles (https://github.com/SUSE/manager-build-profiles) public GitHub repository.

Example of JeOS config.xml.

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

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

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

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

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

8.3.7 Building an Image

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



Build Image

Version:	latest
Image Profile *:	Select an image profile
Build Host *:	Select a build host
Earliest:	05.06.· 18:0 CEST

Profile Summary

No profile selected

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

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

   Note: Profile Summary

   A `Profile Summary` is displayed to the right of the build fields. When you have selected a build profile detailed information about the selected profile will show up in this area.

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

8.3.8 Image Inspection and Salt Integration

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

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

Note

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

The generated pillar is available to all connected minions.

8.3.9 Troubleshooting

Building an image requires of several dependent steps. When the build fails, investigation of salt states results can help you to identify the source of the failure. Usual checks when the build fails:

- The build host can access the build sources
- There is enough disk space for the image on both the build host and the Uyuni server
- The activation key has the correct channels associated with it
- The build sources used are valid
- The RPM package with the Uyuni public certificate is up to date and available at `/usr/share/susemanager/salt/images/rhn-org-trusted-ssl-cert-osimage-1.0-1.noarch.rpm`.
  For more on how to refresh a public certificate RPM, see *Section 8.3.3, "Creating a Build Host"*.

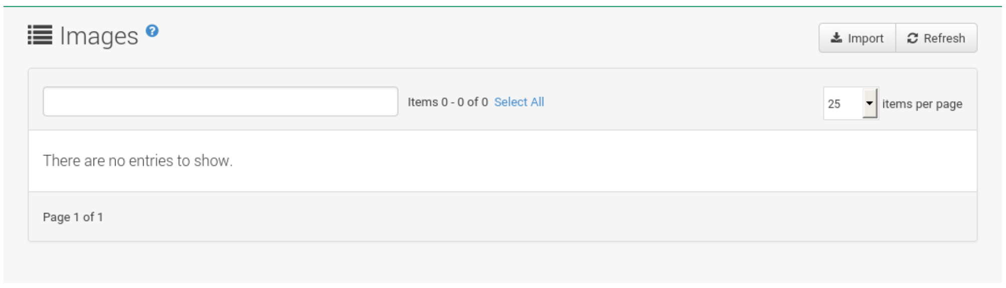
8.3.10 Limitations

The section contains some known issues when working with images.

- HTTPS certificates used to access the HTTP sources or Git repositories should be deployed to the minion by a custom state file, or configured manually.
- Importing Kiwi-based images is not supported.

8.4 Listing Image Profiles Available for Building

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


Images

Items 0 - 0 of 0 Select All

25 items per page

There are no entries to show.

Page 1 of 1

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

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

Note

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

9 Kubernetes Integration Guide

9.1 Prerequisites

The prerequisites listed below should be met before proceeding.

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

  Note

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

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

9.2 Requirements

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

Note

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

9.3 Register Kubernetes as a Virtual Host Manager

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

PROCEDURE: REGISTERING A KUBERNETES CLUSTER WITH UYUNI

1. Select *Systems* › *Virtual Host Managers* from the navigation menu.

2. Expand the `Create` dropdown in the upper right corner of the page and select *Kubernetes Cluster* .

3. Input a label for the new Virtual Host Manager.

4. Select the `kubeconfig` file which contains the required data for the Kubernetes cluster.

5. Select the correct *context* for the cluster, as specified in the kubeconfig file.

6. Click `Create`.

9.4 View the List of Nodes in a Cluster

1. Select *Systems* › *Virtual Host Managers* from the navigation menu.

2. Select the desired Kubernetes cluster to view it.

3. Node data is not refreshed during registration. To refresh node data, click on `Schedule refresh data`.

4. Refresh the browser. If the node data is not available wait a few moments and try again.

9.5 Obtain Runtime Data about Images

See the following steps to find runtime data for images.

1. Select *Images* › *Images* from the navigation menu.

2. In the image list table, take notice of the new runtime columns. These are labeled: `Revision`, `Runtime` and `Instances`. Initially these columns will not provide useful data.

- `Revision`: An artificial sequence number which increments on every rebuild for manager-built images, or on every reimport for externally built images.
- `Runtime`: Overall status of the running instances of the image throughout the registered clusters. The status can be one of the following:
  - All instances are consistent with SUSE Manager: All the running instances are running the same build of the image as tracked by SUSE Manager.
  - Outdated instances found: Some of the instances are running an older build of the image. A redeploy of the image into the pod may be required.
  - No information: The checksum of the instance image does not match the image data contained in SUSE Manager. A redeploy of the image into the pod may be required.
- `Instances`: Number of instances running this image across all the clusters registered in SUSE Manager. A breakdown of numbers can be seen by clicking on the pop-up icon next to the number.

9.6 Build an image for deployment in Kubernetes

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

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

9.7 Import a Previously Deployed Image in Kubernetes

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

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

9.8 Obtain Additional Runtime Data

The following steps will help you find additional runtime data.

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

9.9 Rebuild a Previously Deployed Image in Kubernetes

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

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

9.10 Role Based Access Control Permissions and Certificate Data

Important

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

The API calls from Uyuni are:

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

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

- A ClusterRole to list all the nodes:

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

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

```
resources: ["pods"]
```

```
verbs: ["list"]
```

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

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

10 Cobbler

10.1 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/ ↗.

Important: 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.

10.2 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.

Important: 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.

10.2.1 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.

Note: 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.

10.2.2 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.

10.2.2.1 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; ❶
class "PXE" ❷
{match if substring(option vendor-class-identifier, 0, 9) = "PXEClient"; ❸
next-server 192.168.2.1; ❹
filename "pxelinux.0";} ❺
```

❶ Enable network booting with the `bootp` protocol.

❷ Create a class called `PXE`.

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

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

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

10.2.2.2 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 *Section 10.2.2.1, "Configuring an Existing DHCP Server"*. 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 10.1: 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.

10.2.3 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.

10.2.4 Syncing TFTP Contents to Uyuni Proxies

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

10.2.4.1 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
```

10.2.4.2 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`.

10.2.4.3 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.

Note: 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.

Note: 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.

10.2.4.4 Requirements

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

10.2.5 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
```

Warning

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.

10.3 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 .

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.

10.4 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 .

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 *Section 10.3, "Adding a Distribution to Cobbler"*.

`--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.

10.5 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.

Note

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 .

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 *Section 10.4, “Adding a Profile to Cobbler”*.

`--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`).

10.6 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 .

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.

10.6.1 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.

Note

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

10.6.2 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.

10.7 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.

10.7.1 Using Koan to Provision Virtual Systems

If you have created a virtual machine profile as documented in *Section 10.4, "Adding a Profile to Cobbler"*, 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.

10.7.2 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`.

10.8 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.

Note: 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"
```

Note

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

10.9 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.

10.9.1 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 1 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.

10.9.2 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*.

Note

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.

10.9.3 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.

Note: 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.

10.9.4 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.

Important: 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.

11 Virtualization

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

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

- Autoinstallation is still not supported and libvirt hosts are supported as read-only.

Warning: Limitation to Xen and KVM Guests

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

11.1 Autoinstalling VM Guest s

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

11.1.1 Requirements on Uyuni

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

11.1.1.1 Activation Keys

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

VM HOST SERVERACTIVATION KEY: MINIMUM REQUIREMENTS

- Entitlements: Provisioning, Virtualization Platform.
- Packages: `rhn-virtualization-host` , `osad` .
  If you want to manage the VM Host Server system from Uyuni (e.g. by executing remote scripts), the package `rhncfg-actions` needs to be installed as well.

11.1.1.2 Setting up an Autoinstallable Distribution

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

- Kernel and initrd location:

REDHAT / GENERIC RPM

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

SUSE

- boot/ `arch` /loader/initrd
- boot/ `arch` /loader/linux

    - The *Base Channel* needs to match the autoinstallable distribution.

Important: Autoinstallation package sources

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

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

PROCEDURE: CREATING AUTOINSTALLABLE DISTRIBUTION

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

   Distribution Label

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

   Tree Path

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

   Base Channel

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

   Installer Generation

   Operating system version matching the installation source. This field is mandatory.

   Kernel Options

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

   Post Kernel Options

   Options passed to the kernel when booting the installed system for the first time. This field is optional.
4. Save your settings by clicking *Create Autoinstallable Distribution* .

To edit an existing *Autoinstallable Distribution ›] , go to menu:Systems[Autoinstallation › Distributions* and click on a *Label ›] . Make the desired changes and save your settings by clicking menu:Update Autoinstallable Distribution[.*

11.1.1.3 Providing an Autoinstallation Profile

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

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

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

11.1.1.4 SUSE Linux Enterprise 15 Systems

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

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

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

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

- SLE-Manager-Tools15-Pool
- SLE-Manager-Tools15-Updates

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

It is also recommended to add the following modules:

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

11.1.1.4.1 Uploading an Autoinstallation Profile

1. Log in to the Uyuni Web interface and open *Systems › Autoinstallation › Profiles › Upload New Kickstart/AutoYaST File* .
2. Choose a unique name for the profile. Only letters, numbers, hyphens, periods, and underscores are allowed; the minimum length is 6 characters. This field is mandatory.
3. Choose an *Autoinstallable Tree ›] from the drop-down menu. If no menu:Autoinstallable Tree[* is available, you need to add an Autoinstallable Distribution. Refer to *Section 11.1.1.2, "Setting up an Autoinstallable Distribution"* for instructions.
4. Choose a *Virtualization Type ›] from the drop-down menu. KVM and Xen (para-virtualized and fully-virtualized) are available. Do not choose menu:Xen Virtualized Host[* here.
5. Scroll down to the *File to Upload ›] dialog › click menu:Browse[* to select it, then click *Upload File* .
6. The uploaded file will be displayed in the *File Contents* section, where you can edit it.
7. Click *Create* to store the profile.

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

> **Note: Editing existing {kickstart}profiles**
>
> If you are changing the *Virtualization Type ›] of an existing {kickstart} profile › it may also modify the bootloader and partition options › potentially overwriting any user customizations. Be sure to review the menu:Partitioning[* tab to verify these settings when changing the *Virtualization Type* .

11.1.1.4.2 Creating a Kickstart Profile

Note

Currently it is only possible to create autoinstallation profiles for RHEL systems. If installing a SUSE Linux Enterprise Server system, you need to upload an existing AutoYaST profile as described in *Section 11.1.1.4.1, "Uploading an Autoinstallation Profile"*.

1. Log in to the Uyuni Web interface and go to *Systems* › *Autoinstallation* › *Profiles* › *Create New Kickstart File* .
2. Choose a unique name for the profile. The minimum length is 6 characters. This field is mandatory.
3. Choose a *Base Channel* › *] . This channel is the package source for non-SUSE installations and must match the menu:Autoinstallable Tree[* . This field is mandatory.
4. Choose an *Autoinstallable Tree* › *] from the drop-down menu. If no menu:Autoinstallable Tree[* is available, you need to add an Autoinstallable Distribution. Refer to *Section 11.1.1.2, "Setting up an Autoinstallable Distribution"* for instructions.
5. Choose a *Virtualization Type* › *] from the drop-down menu. KVM and Xen (para-virtualized and fully-virtualized) are available. Do not choose menu:Xen Virtualized Host[* here.
6. Click the *Next* button.
7. Select the location of the distribution files for the installation of your VM Guest s. There should already be a *Default Download Location* › *] filled out and selected for you on this screen. Click the menu:Next[* button.
8. Choose a root password for the VM Guest s. Click the *Finish* button to generate the profile. This completes {kickstart} profile creation. After generating a profile, you are taken to the newly-created {kickstart} profile. You may browse through the various tabs of the profile and modify the settings as you see fit, but this is not necessary as the default settings should work well for the majority of cases.

11.1.1.4.3 Adding a Registration Script to the Autoinstallation Profile

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

1. First, provide a bootstrap script on the Uyuni :
   - Create a bootstrap script for VM Guest s on the Uyuni as described in .
   - Log in as root to the konsole of Uyuni and go to `/srv/www/htdocs/pub/bootstrap` . Copy `bootstrap.sh` (the bootstrap script created in the previous step) to e.g. `bootstrap_vm_guests.sh` in the same directory.
   - Edit the newly created file according to your needs. The minimal requirement is to include the activation key for the VM Guest s (see *Section 11.1.1.1, "Activation Keys"* for details). We strongly recommend to also include one or more GPG keys (for example, your organization key and package signing keys).
2. Log in to the Uyuni Web interface and go to *Systems* › *Autoinstallation* › *Profiles* . Click on the profile that is to be used for autoinstalling the VM Guest s to open it for editing. Scroll down to the *File Contents* section where you can edit the AutoYaST XML file. Add the following snippet at the end of the XML file right before the closing `</profile>` tag and replace the given IP address with the address of the Uyuni server. See *Appendix B, Minimalist AutoYaST Profile for Automated Installations and Useful Enhancements*for an example script.

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

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

3. Click *Update* to save the changes.

11.1.2 VM Host Server Setup

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

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

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

1. Make sure either KVM or Xen is properly set up.
2. Make sure the `libvirtd` is running.
3. Register the VM Host Server with Uyuni :
   - Create a bootstrap script on the Uyuni as described in .
   - Download the bootstrap script from `susemanager.example.com/pub/bootstrap/bootstrap.sh` to the VM Host Server .

- Edit the bootstrap script according to your needs. The minimal requirement is to include the activation key for the VM Host Server (see *Section 11.1.1.1, “Activation Keys”* for details). We strongly recommend to also include one or more GPG keys (for example, your organization key and package signing keys).
- Execute the bootstrap script to register the VM Host Server .

11.1.2.1 VM Host Server setup on Salt clients

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

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

11.1.2.2 VM Host Server setup on Traditional clients

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

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

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

Important: osadTogether with rhnsd

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

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

11.1.3 Autoinstalling VM Guest s

Once all requirements on the Uyuni and the VM Host Server are met, you can start to autoinstall VM Guest s on the host. Note that VM Guest s will not be automatically registered with Uyuni , therefore we strongly recommend to modify the autoinstallation profile as described in *Section 11.1.1.4.3, "Adding a Registration Script to the Autoinstallation Profile"*. VM Guest s need to be registered to manage them with Uyuni . Proceed as follows to autoinstall a VM Guest ;.

Important: No parallel Autoinstallations on VM Host Server

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

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

Note: Checking the Installation Log

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

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

If the event log does not contain enough information to locate a problem, log in to the VM Host Server console and read the log file `/var/log/up2date` . If you are using the `rhnsd` , you may alternatively immediately trigger any scheduled actions by calling `rhn_ckeck` on the VM Host Server . Increase the command's verbosity by using the options `-v`, `-vv`, or `-vvv`, respectively.

11.2 Managing VM Guest s

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

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

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

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

11.2.1 Displaying a VM Guest 's Profile

Click the name of a VM Guest on the VM Host Server 's *Virtualization* page to open its profile page with detailed information about this guest. For details, refer to *Book "Reference Manual", Chapter 7 "Systems"*.

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

11.2.2 Starting, Stopping, Suspending and Resuming a VM Guest

To start, stop, restart, suspend, or resume a VM Guest , navigate to the VM Host Server 's *Virtualization ›] page. Check one or more menu:Guests[* listed in the table and scroll down to the bottom of the page. Choose an action from the drop-down list and click *Apply Action ›] . menu:Confirm[* the action on the next page.

Note: Automatically restarting a VM Guest

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

11.2.3 Changing the CPU or RAM allocation of a VM Guest

To change the CPU or RAM allocation of a VM Guest navigate to the VM Host Server 's *Virtualization ›] page. Check one or more menu:Guests[* from the table and scroll down to the bottom of the page. Choose an action from the *Set ›] drop-down list and provide a new value. Confirm with menu:Apply Changes[* followed by *Confirm* .

The memory allocation can be changed on the fly, provided the memory ballooning driver is installed on the VM Guest . If this is not the case, or if you want to change the CPU allocation, you need to shutdown the guest first. Refer to *Section 11.2.2, "Starting, Stopping, Suspending and Resuming a VM Guest"* for details.

11.2.4 Deleting a VM Guest

To delete a VM Guest you must first shut it down as described in *Section 11.2.2, "Starting, Stopping, Suspending and Resuming a VM Guest"*. Wait at least two minutes to allow the shutdown to finish and then choose *Delete Systems ›] followed by menu:Apply Action[* and *Confirm* .

12 Inventorying vCenter/vSphere ESXi Hosts with Uyuni

12.1 Introduction

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

The following sections will guide you through:

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

12.2 Requirements

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

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

12.3 Permissions and Roles Overview

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

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

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

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

Tip: User Roles/Permissions

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

12.4 Adding New Users and Assigning Roles

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

- Authentication and User Management (https://pubs.vmware.com/vsphere-50/index.jsp#com.vmware.vsphere.security.doc_50/GUID-D7AEC653-EBC8-4573-B990-D8E58742F8ED.html) ↗

12.5 Inventorying vCenter/vSphere ESXi Hosts

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

1. From the Uyuni Web UI select *Main Menu* › *Systems* › *Virtual Host Managers* from the left navigation bar.

2. From the upper right corner of the *Virtual Host Managers* page select *Create* VMWare-based.

3. From the *Add a VMware-based Virtual Host Manager* page complete these fields with your ESXi host data:

   Label
   : Custom name for your Virtual Host Manager

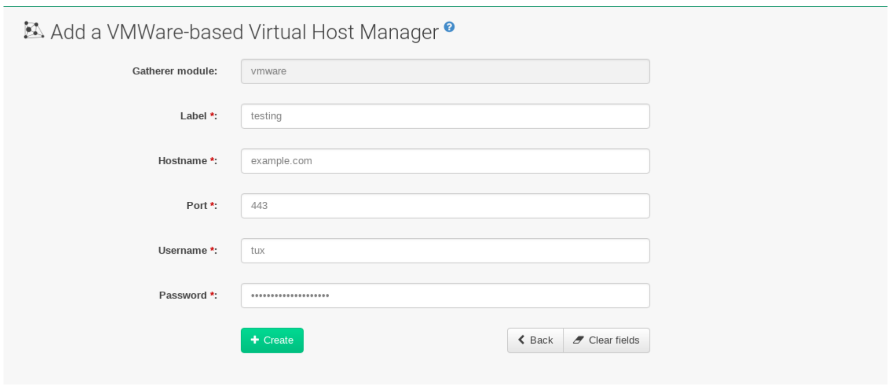
   Hostname
   : Fully-qualified domain name (FQDN) or host IP address

   Port
   : Default ESXi API port

   Username
   : Assign a username

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

   Password
   : ESXi users password



Add a VMWare-based Virtual Host Manager

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

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

Important

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

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

13 SUSE Manager Command Line Tools

13.1 Installing Command Line Tool Packages

In addition to the SUSE Manager Web interface, SUSE Manager offers two command line tools for managing (Traditional) system configuration files :

- The Configuration Client (**mgrcfg-client**)
- The Configuration Manager (**mgrcfg-manager**)

You can use the **mgrcfg-actions** tool to *enable* and *disable* configuration management on client systems.

To work with these tools install them from the SUSE Manager Tools child channel as root.

```
zypper in rhncfg-manager
```

```
zypper in rhncfg-actions
```

Install the following package on client systems:

```
zypper in rhncfg-client
```

Tip: Configuration File Backups

When a configuration file is deployed via SUSE Manager, a backup of the previous file including its full path is stored in the `/var/lib/rhncfg/backups/`. The backup retains its filename but has a *.rhn-cfg-backup* extension appended.

13.2 Actions Control (mgr-actions-control)

The Actions Control (**mgr-actions-control**) application is used to enable and disable configuration management on a system. Client systems cannot be managed in this fashion by default. This tool allows SUSE Manager administrators to enable or disable specific modes of allowable actions such as:

- Deploying a configuration file on the system
- Uploading a file from the system
- Using the diff command to find out what is currently managed on a system with what is available
- Running remote commands

These various modes are enabled/disabled by placing/removing files and directories in the `/etc/sysconfig/rhn/allowed-actions/` directory. Due to the default permissions of the `/etc/sysconfig/rhn/` directory, Actions Control has to be run by someone with root access.

13.2.1 General command line options

There is a manpage available, as for most command line tools. First, decide which scheduled actions should be enabled for use by system administrators. The following options enable the various scheduled action modes:

--enable-deploy

Allow mgrcfg-client to deploy files.

--enable-diff

Allow mgrcfg-client to diff files.

--enable-upload

Allow mgrcfg-client to upload files.

--enable-mtime-upload

Allow mgrcfg-client to upload mtime (file modification time).

--enable-all

Allow mgrcfg-client to do everything.

--enable-run

Enable running scripts.

--disable-deploy

Disable deployment.

--disable-diff

Prohibit diff use.

--disable-upload

No file uploads allowed.

--disable-mtime-upload

Disable mtime upload.

--disable-all

Disable all options.

--disable-run

No scripts allowed to run.

--report

Report whether modes are enabled or disabled.

-f, --force

Force the operation without asking first.

-h, --help

Show help message and exit.

Once a mode is set, your system is ready for configuration management through SUSE Manager. A common option is `mgr-actions-control --enable-all`.

13.3 Configuration Client (mgrcfg-client)

The Configuration Client (mgrcfg-client) is installed on and run from an individual client system to gain knowledge about how SUSE Manager deploys configuration files to the client.

The Configuration Client offers these primary modes:

- list
- get

- channels
- diff
- verify

13.3.1 Listing Configuration Files

To list the configuration files for the machine and the labels of the config channels containing them, issue the command:

```
mgrcfg-client list
```

The output resembles the following list ("DoFoS" is a shortcut for "D or F or S", which means "Directory", "File", or "Something else"(?)):

```
DoFoS   Config Channel      File
F      config-channel-17   /etc/example-config.txt
F      config-channel-17   /var/spool/aalib.rpm
F      config-channel-14   /etc/rhn/rhn.conf
```

These configuration files apply to your system. However, there may be duplicate files present in other channels. For example, issue the following command:

```
mgrcfg-manager list config-channel-14
```

and observe the following output:

```
Files in config channel 'config-channel-14'
/etc/example-config.txt /etc/rhn/rhn.conf
```

You may wonder why the second version of `/etc/example-config.txt` in **config-channel-14** does not apply to the client system. The rank of the `/etc/example-config.txt` file in **config-channel-17** was higher than that of the same file in **config-channel-14**. As a result, the version of the configuration file in config-channel-14 is not deployed for this system, therefore mgrcfg-client command does not list the file.

13.3.2 Downloading a Config File

To download the most relevant configuration file for the machine, issue the command:

```
mgrcfg-client get /etc/example-config.txt
```

You should see output resembling:

```
Deploying /etc/example-config.txt
```

View the contents of the file with less or another pager. Note that the file is selected as the most relevant based on the rank of the config channel containing it. This is accomplished within the Configuration tab of the System Details page.

Refer to Section "System Details" (Chapter 4, Systems, User Guide) for instructions.

13.3.3 Viewing Config Channels

To view the labels and names of the config channels that apply to the system, issue the command:

```
mgrcfg-client channels
```

You should see output resembling:

```
Config channels:
Label                   Name
-----                   ----
config-channel-17       config chan 2
config-channel-14       config chan 1
```

The list of options available for `mgrcfg-client get`:

--topdir=TOPDIR

Make all file operations relative to this string.

--exclude=EXCLUDE

Exclude a file from being deployed with get. May be used multiple times.

-h, --help

Show help message and exit.

13.3.4 Differentiating between Config Files

To view the differences between the config files deployed on the system and those stored by SUSE Manager, issue the command:

```
mgrcfg-client diff
```

The output resembles the following:

```
rhncfg-client diff
--- /etc/test
+++ /etc/test 2013-08-28 00:14:49.405152824 +1000
@@ -1 +1,2 @@
This is the first line
+This is the second line added
```

In addition, you can include the `--topdir` option to compare config files with those located in an arbitrary (and unused) location on the client system, like this:

```
# mgrcfg-client diff --topdir /home/test/blah/
/usr/bin/diff: /home/test/blah/etc/example-config.txt: No such file or directory
/usr/bin/diff: /home/test/blah/var/spool/aalib.rpm: No such file or directory
```

13.3.5 Verifying Config Files

To quickly determine if client configuration files are different from those associated with it via SUSE Manager, issue the command:

```
mgrcfg-client verify
```

The output resembles the following:

```
modified /etc/example-config.txt /var/spool/aalib.rpm
```

The file `example-config.txt` is locally modified, while `aalib.rpm` is not.

The list of the options available for mgrcfg-client verify:

-v, --verbose

Increase the amount of output detail. Display differences in the mode, owner, and group permissions for the specified config file.

-o, --only

Only show differing files.

-h, --help

Show help message and exit.

13.4 Configuration Manager (mgrcfg-manager)

The Configuration Manager (**mgrcfg-manager**) is designed to maintain SUSE Manager's central repository of config files and channels, not those located on client systems. This tool offers a command line alternative to the configuration management features in the SUSE Manager Web interface. Additionally, some or all of the related maintenance tasks can be scripted.

To use the command line interface, configuration administrators require a SUSE Manager account (username and password) with the appropriate permission set. The username may be specified in `/etc/sysconfig/rhn/rhncfg-manager.conf` or in the `[rhncfg-manager]` section of `~/.rhncfgrc`.

When the Configuration Manager is run as **root**, it attempts to pull in needed configuration values from the Red Hat Update Agent. When run as a user other than root, you may have to change the `~/.rhncfgrc` configuration file. The session file is cached in `~/.rhncfg-manager-session` to avoid having to log in for every command.

The default timeout for the Configuration Manager is 30 minutes. To adjust this, add the `server.session_lifetime` option and a new value to the `/etc/rhn/rhn.conf` file on the server running the manager. For example set the time out to **120 minutes**:

```
server.session_lifetime = 120
```

The Configuration Manager offers the following primary **modes**:

- add
- create-channel
- diff
- diff-revisions
- download-channel
- get
- list
- list-channels

- remove
- remove-channel
- revisions
- update
- upload-channel

Each mode offers its own set of options, which can be displayed by issuing the following command:

```
mgrcfg-manager mode --help
```

Replace mode with the name of the mode whose options you want to see:

```
mgrcfg-manager diff-revisions --help
```

13.4.1 Creating a Config Channel

To create a config channel for your organization, issue the command:

```
mgrcfg-manager create-channel channel-label
```

If prompted for your SUSE Manager username and password, provide them. Once you have created a config channel, use the remaining modes listed above to populate and maintain that channel.

13.4.2 Adding Files to a Config Channel

To add a file to a config channel, specify the channel label and the local file to be uploaded:

```
mgrcfg-manager add --channel=channel-label /path/to/file
```

In addition to the required channel label and the path to the file, you can use the available options for modifying the file during its addition. For instance, you can alter the path and file name by including the `--dest-file` option in the command:

```
mgrcfg-manager add --channel=channel-label \
  --dest-file=/new/path/to/file.txt/path/to/file
```

The output resembles the following:

```
Pushing to channel example-channel
Local file >/path/to/file -> remote file /new/path/to/file.txt
```

The list of options available for mgrcfg-manager add:

-c CHANNEL --channel=CHANNEL
: Upload files in this config channel.

-d DEST_FILE --dest-file=DEST_FILE
: Upload the file as this path.

--delim-start=DELIM_START
: Start delimiter for variable interpolation.

--delim-end=DELIM_END
: End delimiter for variable interpolation.

-i, --ignore-missing
: Ignore missing local files.

-h, --help
: Show help message and exit.

Tip: Max File size

By default, the maximum file size for configuration files is 128KB. If you need to change that value, find or create the following line in the `/etc/rhn/rhn.conf` file:

```
web.maximum_config_file_size=128
```

Change the value from 128 to whatever limit you need in kilobytes.

13.4.3 Differentiating between Latest Config Files

To view the differences between the config files on disk and the latest revisions in a channel, issue the command:

```
mgrcfg-manager diff --channel=channel-label --dest-file=/path/to/file.txt \
/local/path/to/file
```

You should see output resembling:

```
--- /tmp/dest_path/example-config.txt config_channel: example-channel revision: 1
+++ /home/test/blah/hello_world.txt 2003-12-14 19:08:59.000000000 -0500
@@ -1 +1 @@
-foo
+hello, world
```

The list of options available for `mgrcfg-manager diff`:

-c CHANNEL, --channel=CHANNEL

Get file(s) from this config channel.

-r REVISION, --revision=REVISION

Use this revision.

-d DEST_FILE, --dest-file=DEST_FILE

Upload the file at this path.

-t TOPDIR, --topdir=TOPDIR

Make all files relative to this string.

-h, --help

Show help message and exit.

13.4.4 Differentiating between Various Versions

To compare different versions of a file across channels and revisions, use the **-r** flag to indicate which revision of the file should be compared and the **-n** flag to identify the two channels to be checked. Specify only one file name here since you are comparing the file against another version of itself. For example:

```
mgrcfg-manager diff-revisions -n=channel-label1 -r=1 \
  -n=channel-label2 -r=1 \
  /path/to/file.txt
```

The output resembles the following:

```
--- /tmp/dest_path/example-config.txt 2004-01-13 14:36:41 \
config channel: example-channel2 revision: 1
--- /tmp/dest_path/example-config.txt 2004-01-13 14:42:42 \
config channel: example-channel3 revision: 1
```

```
@@ -1 +1,20 @@
-foo
+blah
+-----BEGIN PGP SIGNATURE-----
+Version: GnuPG v1.0.6 (GNU/Linux)
+Comment: For info see http://www.gnupg.org
+
+iD8DBQA9ZY6vse4XmfJPGwgRAsHcAJ9ud9dabUcdscdcqB8AZP7e0Fua0NmKsdhQCeOWHX
+VsDTfen2NWdwwPaTM+S+Cow=
+=Ltp2
+-----END PGP SIGNATURE-----
```

The list of options available for `mgrcfg-manager diff-revisions`:

-c CHANNEL, --channel=CHANNEL
: Use this config channel.

-r REVISION, --revision=REVISION
: Use this revision.

-h, --help
: Show help message and exit.

13.4.5 Downloading All Files in a Channel

To download all the files in a channel to disk, create a directory and issue the following command:

```
mgrcfg-manager download-channel channel-label --topdir .
```

The output resembles the following:

```
Copying /tmp/dest_path/example-config.txt -> \
blah2/tmp/dest_path/example-config.txt
```

The list of options available for mgrcfg-manager download-channel:

-t TOPDIR, --topdir=TOPDIR
: Directory to which all the file paths are relative. This option must be set.

-h, --help
: Show help message and exit.

13.4.6 Getting the Contents of a File

To direct the contents of a particular file to stdout, issue the command:

```
mgrcfg-manager get --channel=channel-label \
/tmp/dest_path/example-config.txt
```

You should see the contents of the file as the output.

13.4.7 Listing All Files in a Channel

To list all the files in a channel, issue the command:

```
mgrcfg-manager list channel-label
```

You should see output resembling:

```
Files in config channel `example-channel3':
/tmp/dest_path/example-config.txt
```

The list of the options available for mgrcfg-manager get:

-c CHANNEL, --channel=CHANNEL
: Get file(s) from this config channel.

-t TOPDIR, --topdir=TOPDIR
: Directory to which all files are relative.

-r REVISION, --revision=REVISION
: Get this file revision.

-h, --help
: Show help message and exit.

13.4.8 Listing All Config Channels

To list all of your organization’s configuration channels, issue the command:

```
mgrcfg-manager list-channels
```

The output resembles the following:

```
Available config channels:
example-channel example-channel2 example-channel3 config-channel-14 config-channel-17
```

Note

This does not list **local_override** or **server_import** channels.

13.4.9 Removing a File from a Channel

To remove a file from a channel, issue the command:

```
mgrcfg-manager remove --channel=channel-label /tmp/dest_path/example-config.txt
```

If prompted for your SUSE Manager username and password, provide them.

The list of the options available for mgrcfg-manager remove:

-c CHANNEL, --channel=CHANNEL

Remove files from this config channel.

-t TOPDIR, --topdir=TOPDIR

Directory to which all files are relative.

-h, --help

Show help message and exit.

13.4.10 Deleting a Config Channel

To remove a config channel in your organization, issue the command:

```
mgrcfg-manager remove-channel channel-label
```

The output resembles the following:

```
Removing config channel example-channel
Config channel example-channel removed
```

13.4.11 Determining the Number of File Revisions

To find out how many revisions (from **1 to N** where N is an integer greater than 0) of a file/path are in a channel, issue the following command:

```
mgrcfg-manager revisions channel-label /tmp/dest_path/example-config.txt
```

The output resembles the following:

```
Analyzing files in config channel example-channel \
/tmp/dest_path/example-config.txt: 1
```

13.4.12 Updating a File in a Channel

To create a new revision of a file in a channel (or to add the first revision to that channel if none existed before for the given path), issue the following command:

```
mgrcfg-manager update --channel=channel-label \
--dest-file=/path/to/file.txt /local/path/to/file
```

The output resembles the following:

```
Pushing to channel example-channel:
Local file example-channel /tmp/local/example-config.txt -> \
remote file /tmp/dest_path/example-config.txt
```

The list of the options available for mgrcfg-manager update:

-c CHANNEL, --channel=CHANNEL
: Upload files in this config channel.

-d DEST_FILE, --dest-file=DEST_FILE
: Upload the file to this path.

-t TOPDIR, --topdir=TOPDIR
: Directory to which all files are relative.

--delim-start=DELIM_START
: Start delimiter for variable interpolation.

--delim-end=DELIM_END
: End delimiter for variable interpolation.

-h, --help

Show help message and exit.

13.4.13 Uploading Multiple Files at Once

To upload multiple files to a config channel from a local disk at once, issue the command:

```
mgrcfg-manager upload-channel --topdir=topdir channel-label
```

The output resembles the following:

```
Using config channel example-channel4
Uploading /tmp/ola_world.txt from blah4/tmp/ola_world.txt
```

The list of the options available for mgrcfg-manager upload-channel:

-t TOPDIR, --topdir=TOPDIR

Directory all the file paths are relative to.

-c CHANNEL, --channel=CHANNEL

List of channels the config info will be uploaded into channels delimited by ','. Example: --channel=foo,bar,baz.

-h, --help

Show help message and exit.

13.5 Syncing SUSE Manager Repositories from SCC (mgr-sync)

`mgr-sync` should be used if SUSE Manager is connected to SUSE Customer Center (SCC). With `mgr-sync` you may add or synchronize products and channels. The `mgr-sync` command also enables and refreshes SCC data.

This tool requires that SCC is enabled by running `mgr-sync enable-scc` first (Enabled by default in Uyuni 2.1 and greater).

By default, log rotation of `spacewalk-repo-sync` is disabled. This logfile is run by a cron job, and managed by a configuration file. You can enable log rotation by editing the configuration file at `etc/sysconfig/rhn/reposync` and setting the `MAX_DAYS` parameter. This will permit the deletion of files that are older than the specified period of time.

Important: Admin credentials

`mgr-sync` now requires the username/password of a **SUSE Manager administrator**. Most functions are available as part of the public API.

mgr-sync provides a command structure with sub-commands similar to git or osc. For a complete list of command line option, see the mgr-sync manpage (man mgr-sync). Basic actions are:

```
mgr-sync enable-scc

mgr-sync list channel(s)|product(s)|credentials
mgr-sync add  channel(s)|product(s)|credentials
mgr-sync delete  credentials

mgr-sync refresh [--refresh-channels] [--from-mirror MIRROR]
```

See the following examples.

List channels

```
mgr-sync list channels
```

Add a channel

```
mgr-sync add channel LABEL
```

List products

```
mgr-sync list products
```

Add a product

```
mgr-sync add product
```

Refresh the data

```
mgr-sync refresh
```

Refresh data and schedule a reposync for all installed vendor channels

```
mgr-sync refresh --refresh-channels
```

List SCC credentials

```
mgr-sync list credentials
```

Add new SCC credentials

```
mgr-sync add credentials
```

Tip: Credentials

There can be one primary credential only. This is username/password used first when retrieving the list of available channels and packages.

Add SCC primary credentials

```
mgr-sync add credentials --primary
```

Delete SCC credentials

```
mgr-sync delete credentials
```

13.6 Configuring SUSE Manager's Database (smdba)

SUSE Manager provides the smdba command for managing the installed database. It is the successor of `db-control`, which is now **unsupported**.

The smdba command works on local databases only, not remote. This utility allows you to do several administrative tasks like backing up and restoring the database. It also allows you to create, verify, restore backups, obtaining database status, and restart the database if necessary. The smdba command supports **PostgreSQL**.

Find basic information about **smdba** in the **smdba manpage**.

Note: Restart Spacewalk Services When Connection is Lost

If you have stopped or restarted the database, Spacewalk services can lose their connections. In such a case, run the following command:

```
spacewalk-service restart
```

13.6.1 Control Options

Depending on the database installed, smdba provides several subcommands:

```
backup-hot      Enable continuous archiving backup
backup-restore  Restore the SUSE Manager Database from backup.
backup-status   Show backup status.
db-start        Start the SUSE Manager Database.
db-status       Show database status.
db-stop         Stop the SUSE Manager Database.
space-overview  Show database space report.
space-reclaim   Free disk space from unused object in tables and indexes.
space-tables    Show space report for each table.
system-check    Common backend healthcheck.
```

For a list of available commands on your particular appliance, call smdba help. To display the help message for a specific subcommand, call `smdba COMMAND help`.

13.6.2 Starting and Stopping the Database

There are three commands to start, stop, or get the status of the database. Use the following commands:

```
# smdba db-status
Checking database core...       online
# smdba db-stop
Stopping the SUSE Manager database...
Stopping listener:     done
Stopping core:         done
# smdba db-status
Checking database core...       offline
# smdba db-start
Starting listener:     done
Starting core...       done
```

13.7 Creating a Bootstrap Repository (mgr-create-bootstrap-repo)

The `mgr-create-bootstrap-repo` command is used on the Uyuni Server to create a new bootstrap repository.

Use the `-l` option to list all available repositories:

```
# mgr-create-bootstrap-repo -l
```

You can then invoke the command with the appropriate repository name to create the bootstrap repository you require, for example:

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

```
[[at.clitools.createbootstraprepo.customchannels]]
=== Creating a Bootstrap Repository with Custom Channels
```

```
Custom channels are channels that have been created to manage any custom packages that an
 organization might require.
To create a new bootstrap repository from a custom channel, use the [code]``mgr-create-
bootstrap-repo`` command with the [code]``with-custom-channels`` option:
```

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

Note: Flushing a Bootstrap Repository to Remove Custom Channels

If you create a bootstrap repository that contains custom channels, and later attempt to rebuild with the `mgr-create-bootstrap-repo` command, the custom channel information will remain in the bootstrap repository. If you want to remove custom channel information from your bootstrap repository, you will need to use the `flush` option when you rebuild:

```
# mgr-create-bootstrap-repo --flush
```

14 spacecmd Reference

14.1 Introduction

The following section will help you become more familiar with the `spacecmd` command-line interface. This interface is available for SUSE Manager, Satellite and Spacewalk servers. spacecmd is written in Python and uses the XML-RPC API provided by the server.

WHAT CAN SPACECMD DO FOR ME?

- Manage almost all aspects of SUSE Manager from the command line with spacecmd
- Tab completion is available for all commands
- Single commands can be passed to spacecmd without entering the interactive shell (excellent for shell scripts)
- May also be accessed and used as an interactive shell
- Advanced search methods are available for finding specific systems, thus removing the need to create system groups (nevertheless groups are still recommended)
- Complete functionality through the Spacewalk API. Almost all commands that can be executed from the WebUI can be performed via the spacecmd command-line

14.2 Configuring spacecmd

The following section provides configuration tips for spacecmd.

14.2.1 Setup Spacecmd Credentials

Normally spacecmd prompts you for a username and password each time you attempt to login to the interactive shell. Alternatively you can configure spacecmd with a credentials file to avoid this requirement.

PROCEDURE: CREATING A SPACECMD CREDENTIALS FILE

1. Create a hidden spacecmd directory in your home directory and set permissions:

```
mkdir ~/.spacecmd
```

```
chmod 700 ~/.spacecmd
```

2. Create a `config` file in `~/.spacecmd/` and provide proper permissions:

```
touch ~/.spacecmd/config
chmod 600 ~/.spacecmd/config
```

3. Edit the `config` file and add the following configuration lines. (You can use either localhost or the FQDN of your SUSE Manager server):

```
[spacecmd]

server=FQDN-here

username=username-here
password=password-here
```

4. Check connectivity by entering `spacecmd` as root:

```
# spacecmd
```

14.2.2 spacecmd Quiet Mode

By default spacecmd prints server status messages during connection attempts. These messages can cause a lot of clutter when parsing system lists. The following alias will force spacecmd to use quiet mode thus preventing this behavior. Add the following line to your `~/.bashrc` file:

```
alias spacecmd='spacecmd -q'
```

14.2.3 spacecmd Help

spacecmd help can be access by typing spacecmd `-h --help`

```
Usage: spacecmd [options] [command]

Options:
  -u USERNAME, --username=USERNAME
                        use this username to connect to the server
  -p PASSWORD, --password=PASSWORD
                        use this password to connect to the server
  -s SERVER, --server=SERVER
                        connect to this server [default: local hostname]
  --nossl               use HTTP instead of HTTPS
  --nohistory           do not store command history
  -y, --yes             answer yes for all questions
  -q, --quiet           print only error messages
  -d, --debug           print debug messages (can be passed multiple times)
  -h, --help            show this help message and exit
```

As root you can access available functions without entering the spacecmd shell:

```
# spacecmd -- help

        Documented commands (type help <topic>):
========================================
activationkey_addchildchannels       org_trustdetails
activationkey_addconfigchannels      package_details
activationkey_addentitlements        package_listdependencies
activationkey_addgroups              package_listerrata
activationkey_addpackages            package_listinstalledsystems
activationkey_clone                  package_listorphans
activationkey_create                 package_remove
activationkey_delete                 package_removeorphans
activationkey_details                package_search
activationkey_diff                   repo_addfilters
activationkey_disable                repo_clearfilters
activationkey_disableconfigdeployment  repo_create

...
```

14.3 Troubleshooting

This section provides troubleshooting solutions when working with spacecmd

14.3.1 Creating a Distribution With spacecmd Sets Localhost Instead of FQDN

The support article associated with this issue may be located at:https://www.suse.com/support/kb/doc/?id=7018627

Situation

When creating a distribution with spacecmd it will automatically set localhost as the server name instead of the FQDN of SUSE Manager. This will result in the following kernel option being written:

```
install=http://localhost/ks/dist/<distributionname>
```

Resolution

Set the FQDN in `$HOME/.spacecmd/config` like the following:

```
test:~/.spacecmd # cat config
[spacecmd]
server=test.mytest.env
username=admin
password=password
nossl=0
```

Cause

This problem may be experienced if `$HOME/.spacecmd/config` has been created and the server name option was set to localhost.

14.3.2 Spacecmd not Accepting Commands or Options

When running `spacecmd` non-interactively, you must escape arguments passed to the command. Always put `--` before arguments, to avoid them being treated as global arguments. Additionally, make sure you escape any quotes that you pass to the functions so that they are not interpreted. An example of a well-formed `spacecmd` command:

```
spacecmd -s server1 -- softwarechannel_create -n \'My Channel\' -l channel1 -a x86_64
```

14.4 spacecmd Functions

The following sections provide descriptions for all documented spacecmd commands. Each command is grouped by the function prefix. Keep in mind that all commands may also be called using scripts and passed to spacecmd as stand-alone commands.

14.4.1 activationkey_

The following spacecmd commands are available for use with activation keys.

activationkey_addchildchannels

Add child channels to an activation key.

```
usage: activationkey_addchildchannels KEY <CHANNEL ...>
```

activationkey_addconfigchannels

Add configuration channels to an activation key.

```
usage: activationkey_addconfigchannels KEY <CHANNEL ...> [options]

options:
  -t add channels to the top of the list
  -b add channels to the bottom of the list
```

activationkey_addentitlements

Add available entitlements to an activation key.

Note: WebUI Name Change

In the WebUI entitlements are known as System Types. Nevertheless the spacecmd backend still utilizes the entitlements term. Therefore any scripts you may be using can remain unchanged.

```
usage: activationkey_addentitlements KEY <ENTITLEMENT ...>
```

activationkey_addgroups

Add existing groups to an activation key.

```
usage: activationkey_addgroups KEY <GROUP ...>
```

activationkey_addpackages

Add packages to an activation key.

```
usage: activationkey_addpackages KEY <PACKAGE ...>
```

activationkey_clone

Clone an existing activation key.

```
usage examples:
                 activationkey_clone foo_key -c bar_key
                 activationkey_clone foo_key1 foo_key2 -c prefix
                 activationkey_clone foo_key -x "s/foo/bar"
                 activationkey_clone foo_key1 foo_key2 -x "s/foo/bar"

options:
  -c CLONE_NAME  : Name of the resulting key, treated as a prefix for multiple
                   keys
  -x "s/foo/bar" : Optional regex replacement, replaces foo with bar in the
                   clone description, base-channel label, child-channel
                   labels, config-channel names
```

activationkey_create

Create a new activation key.

```
usage: activationkey_create [options]

options:
  -n NAME
  -d DESCRIPTION
  -b BASE_CHANNEL
  -u set key as universal default
  -e [enterprise_entitled,virtualization_host]
```

activationkey_delete

Delete an existing activation key.

```
usage: activationkey_delete KEY
```

activationkey_details

Show details of an existing activation key.

```
usage: activationkey_details KEY ...
```

activationkey_diff

Check the difference between two activation keys.

```
usage: activationkey_diff SOURCE_ACTIVATIONKEY TARGET_ACTIVATIONKEY
```

activationkey_disable

Disable an existing activation key.

```
usage: activationkey_disable KEY [KEY ...]
```

activationkey_disableconfigdeployment

Disable configuration channel deployment for an existing activation key.

```
usage: activationkey_disableconfigdeployment KEY
```

activationkey_enable

Enable an existing activation key.

```
usage: activationkey_enable KEY [KEY ...]
```

activationkey_enableconfigdeployment

Enable configuration channel deployment for an existing activation key.

```
usage: activationkey_enableconfigdeployment KEY
```

activationkey_export

Export activation key(s) to a JSON formatted file.

```
usage: activationkey_export [options] [<KEY> ...]

options:
    -f outfile.json : specify an output filename, defaults to <KEY>.json
                      if exporting a single key, akeys.json for multiple keys,
                      or akey_all.json if no KEY specified (export ALL)

Note : KEY list is optional, default is to export ALL keys
```

activationkey_import

Import activation key(s) from JSON file(s)

```
usage: activationkey_import <JSONFILE ...>
```

activationkey_list

List all existing activation keys.

```
usage: activationkey_list
```

activationkey_listbasechannel

List the base channel associated with an activation key.

```
usage: activationkey_listbasechannel KEY
```

activationkey_listchildchannels

List child channels associated with an activation key.

```
usage: activationkey_listchildchannels KEY
```

activationkey_listconfigchannels

List configuration channels associated with an activation key.

```
usage: activationkey_listconfigchannels KEY
```

activationkey_listentitlements

List entitlements associated with an activation key.

```
usage: activationkey_listentitlements KEY
```

activationkey_listgroups

List groups associated with an activation key

```
usage: activationkey_listgroups KEY
```

activationkey_listpackages

List packages associated with an activation key.

```
usage: activationkey_listpackages KEY
```

activationkey_listsystems

List systems registered with an activation key.

```
usage: activationkey_listsystems KEY
```

activationkey_removechildchannels

Remove child channels from an activation key.

```
usage: activationkey_removechildchannels KEY <CHANNEL ...>
```

activationkey_removeconfigchannels

Remove configuration channels from an activation key.

```
usage: activationkey_removeconfigchannels KEY <CHANNEL ...>
```

activationkey_removeentitlements

Remove entitlements from an activation key.

```
usage: activationkey_removeentitlements KEY <ENTITLEMENT ...>
```

activationkey_removegroups

Remove groups from an activation key.

```
usage: activationkey_removegroups KEY <GROUP ...>
```

activationkey_removepackages

Remove packages from an activation key.

```
usage: activationkey_removepackages KEY <PACKAGE ...>
```

activationkey_setbasechannel

Set the base channel for an activation key.

```
usage: activationkey_setbasechannel KEY CHANNEL
```

activationkey_setconfigchannelorder

Set the ranked order of configuration channels.

```
usage: activationkey_setconfigchannelorder KEY
```

activationkey_setcontactmethod

Set the contact method to use for systems registered with a specific key. (Use the XML-RPC API to access the latest contact methods.) The following contact methods are available for use with traditional spacecmd: ['default', 'ssh-push', 'ssh-push-tunnel']

```
usage: activationkey_setcontactmethod KEY CONTACT_METHOD
```

activationkey_setdescription

Add a description for an activation key.

```
usage: activationkey_setdescription KEY DESCRIPTION
```

activationkey_setuniversaldefault

Set a specific key as the universal default.

```
usage: activationkey_setuniversaldefault KEY
```

Warning: Universal Default Key

Using a universal default key is not a Best Practice recommendation.

activationkey_setusagelimit

Set the usage limit of an activation key, can be a number or "unlimited".

```
usage: activationkey_setbasechannel KEY <usage limit>
usage: activationkey_setbasechannel KEY unlimited
```

Tip: Usage Limits

Usage limits are only applicable to traditionally managed systems. Currently usage limits do not apply to Salt or foreign managed systems.

14.4.2 api

The following API command and its options are available for calling the XML-RPC API directly. Calling the API directly allows you to use the latest features in SUSE Manager from the command-line using spacecmd as a wrapper for stand-alone commands or used from within scripts.

Note: Use the api Command for Access to Latest Features

spacecmd is the traditional tool for spacewalk. It functions out of the box with SUSE Manager but you should know that latest features (for example, Salt) are often excluded from traditional spacecmd command-line tool. To gain access to the latest feature additions call `api api.getApiCallList` from within spacecmd to list all currently available API commands formatted in json. You can then call these commands directly.

api

Call XML-RPC API with arguments directly.

```
usage: api [options] API_STRING

options:
  -A, --args  Arguments for the API other than session id in comma separated
              strings or JSON expression
  -F, --format   Output format
  -o, --output   Output file

examples:
  api api.getApiCallList
  api --args "sysgroup_A" systemgroup.listSystems
  api -A "rhel-i386-server-5,2011-04-01,2011-05-01" -F "%(name)s" \
      channel.software.listAllPackages
```

14.4.3 clear

Clears the terminal screen

14.4.4 clear_caches

Clear the internal caches kept for systems and packages

```
usage: clear_caches
```

14.4.5 configchannel_

The following spacecmd commands are available for use with configuration channels.

configchannel_addfile

Creates a configuration file.

```
usage: configchannel_addfile [CHANNEL] [options]

options:
  -c CHANNEL
  -p PATH
  -r REVISION
  -o OWNER [default: root]
  -g GROUP [default: root]
  -m MODE [defualt: 0644]
  -x SELINUX_CONTEXT
  -d path is a directory
  -s path is a symlink
  -b path is a binary (or other file which needs base64 encoding)
  -t SYMLINK_TARGET
  -f local path to file contents

  Note re binary/base64: Some text files, notably those containing trailing
  newlines, those containing ASCII escape characters (or other charaters not
  allowed in XML) need to be sent as binary (-b).  Some effort is made to auto-
  detect files which require this, but you may need to explicitly specify.
```

configchannel_backup

Backup a configuration channel.

```
usage: configchannel_backup CHANNEL [OUTDIR]

OUTDIR defaults to $HOME/spacecmd-backup/configchannel/YYYY-MM-DD/CHANNEL
```

configchannel_clone

Clone configuration channel(s).

```
usage examples:
                 configchannel_clone foo_label -c bar_label
                 configchannel_clone foo_label1 foo_label2 -c prefix
                 configchannel_clone foo_label -x "s/foo/bar"
                 configchannel_clone foo_label1 foo_label2 -x "s/foo/bar"

options:
  -c CLONE_LABEL : name/label of the resulting cc (note does not update
                   description, see -x option), treated as a prefix if
                   multiple keys are passed
  -x "s/foo/bar" : Optional regex replacement, replaces foo with bar in the
                   clone name, label and description
  Note : If no -c or -x option is specified, interactive is assumed
```

configchannel_create

Create a configuration channel.

```
usage: configchannel_create [options]

options:
  -n NAME
  -l LABEL
  -d DESCRIPTION
```

configchannel_delete

Delete a configuration channel.

```
usage: configchannel_delete CHANNEL ...
```

configchannel_details

Show the details of a configuration channel.

```
usage: configchannel_details CHANNEL ...
```

configchannel_diff

Find differences between configuration channels.

```
usage: configchannel_diff SOURCE_CHANNEL TARGET_CHANNEL
```

configchannel_export

Export configuration channel(s) to a json formatted file.

```
usage: configchannel_export <CHANNEL>... [options]
options:
    -f outfile.json : specify an output filename, defaults to <CHANNEL>.json
                      if exporting a single channel, ccs.json for multiple
                      channels, or cc_all.json if no CHANNEL specified
                      e.g (export ALL)

Note : CHANNEL list is optional, default is to export ALL
```

configchannel_filedetails

Show the details of a file in a configuration channel.

```
usage: configchannel_filedetails CHANNEL FILE [REVISION]
```

configchannel_forcedeploy

Forces a redeployment of files within a channel on all subscribed systems.

```
usage: configchannel_forcedeploy CHANNEL
```

configchannel_import

Import configuration channel(s) from a json file.

```
usage: configchannel_import <JSONFILES...>
```

configchannel_list

List all configuration channels.

```
usage: configchannel_list
```

configchannel_listfiles

List all files in a configuration channel.

```
usage: configchannel_listfiles CHANNEL ...
```

configchannel_listsystems

List all systems subscribed to a configuration channel.

```
usage: configchannel_listsystems CHANNEL
```

configchannel_removefiles

Remove configuration files.

```
usage: configchannel_removefile CHANNEL <FILE ...>
```

configchannel_sync

Sync configuration files between two configuration channels.

```
usage: configchannel_sync SOURCE_CHANNEL TARGET_CHANNEL
```

configchannel_updatefile

Update a configuration file.

```
usage: configchannel_updatefile CHANNEL FILE
```

configchannel_verifyfile

Verify a configuration file.

```
usage: configchannel_verifyfile CHANNEL FILE <SYSTEMS>

<SYSTEMS> may be substituted with any of the following targets:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

14.4.6 cryptokey_

The following spacecmd commands are available for use with cryptographic keys.

cryptokey_create

Create a cryptographic key.

```
usage: cryptokey_create [options]

options:
  -t GPG or SSL
  -d DESCRIPTION
  -f KEY_FILE
```

cryptokey_delete

Delete a cryptographic key.

```
usage: cryptokey_delete NAME
```

cryptokey_details

Show the contents of a cryptographic key.

```
usage: cryptokey_details KEY ...
```

cryptokey_list

List all cryptographic keys (SSL, GPG).

```
usage: cryptokey_list
```

14.4.7 custominfo_

The following spacecmd commands are available for working with custom keys.

custominfo_createkey

Create a custom key.

```
usage: custominfo_createkey [NAME] [DESCRIPTION]
```

custominfo_deletekey

Delete a custom key.

```
usage: custominfo_deletekey KEY ...
```

custominfo_details

Show the details of a custom key.

```
usage: custominfo_details KEY ...
```

custominfo_listkeys

List all custom keys.

```
usage: custominfo_listkeys
```

custominfo_updatekey

Update a custom key.

```
usage: custominfo_updatekey [NAME] [DESCRIPTION]
```

14.4.8 distribution_

The following spacecmd commands are available for working with kickstart distributions.

distribution_create

Create a Kickstart tree.

```
usage: distribution_create [options]

options:
  -n NAME
  -p path to tree
  -b base channel to associate with
  -t install type [fedora|rhel_4/5/6|suse|generic_rpm]
```

distribution_delete

Delete a Kickstart tree.

```
usage: distribution_delete LABEL
```

distribution_details

Show the details of a Kickstart tree.

```
usage: distribution_details LABEL
```

distribution_list

List the available autoinstall trees.

```
usage: distribution_list
```

distribution_rename

Rename a Kickstart tree.

```
usage: distribution_rename OLDNAME NEWNAME
```

distribution_update

Update the path of a Kickstart tree.

```
usage: distribution_update NAME [options]

options:
  -p path to tree
  -b base channel to associate with
  -t install type [fedora|rhel_4/5/6|suse|generic_rpm]
```

14.4.9 errata_

The following spacecmd commands are available for use with errata data.

errata_apply

Apply an patch to all affected systems.

```
usage: errata_apply ERRATA|search:XXX ...
```

errata_delete

Delete an patch.

```
usage: errata_delete ERRATA|search:XXX ...
```

errata_details

Show the details of an patch.

```
usage: errata_details ERRATA|search:XXX ...
```

errata_findbycve

List errata addressing a CVE.

```
usage: errata_findbycve CVE-YYYY-NNNN ...
```

errata_list

List all patches.

```
usage: errata_list
```

errata_listaffectedsystems

List of systems affected by an patch.

```
usage: errata_listaffectedsystems ERRATA|search:XXX ...
```

errata_listcves

List of CVEs addressed by an patch.

```
usage: errata_listcves ERRATA|search:XXX ...
```

errata_publish

Publish an patch to a channel.

```
usage: errata_publish ERRATA|search:XXX <CHANNEL ...>
```

errata_search

List patches that meet user provided criteria

```
usage: errata_search CVE|RHSA|RHBA|RHEA|CLA ...

Example:
> errata_search CVE-2009:1674
> errata_search RHSA-2009:1674
```

errata_summary

Print a summary of all errata.

```
usage: errata_summary
```

14.4.10 filepreservation_

The following spacecmd commands are available for working with kickstart file preservation lists.

filepreservation_create

Create a file preservation list.

```
usage: filepreservation_create [NAME] [FILE ...]
```

filepreservation_delete

Delete a file preservation list.

```
filepreservation_delete NAME
```

filepreservation_details

Show the details of a file preservation list.

```
usage: filepreservation_details NAME
```

filepreservation_list

List all file preservations.

```
usage: filepreservation_list
```

14.4.11 get_

get_apiversion

Display the API version of the server.

```
usage: get_apiversion
```

get_certificateexpiration

Print the expiration date of the server’s entitlement certificate.

```
usage: get_certificateexpiration
```

get_serverversion

Display SUSE Manager server version.

```
usage: get_serverversion
```

get_session

Show the current session string.

```
usage: get_session
```

14.4.12 group_

group_addsystems

Add systems to a group.

```
usage: group_addsystems GROUP <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

group_backup

Backup a system group.

```
usage: group_backup NAME [OUTDIR]

OUTDIR defaults to $HOME/spacecmd-backup/group/YYYY-MM-DD/NAME
```

group_create

Create a system group.

```
usage: group_create [NAME] [DESCRIPTION]
```

group_delete

Delete a system group.

```
usage: group_delete NAME ...
```

group_details

Show the details of a system group.

```
usage: group_details GROUP ...
```

group_list

List available system groups.

```
usage: group_list
```

group_listsystems

List the members of a group.

```
usage: group_listsystems GROUP
```

group_removesystems

Remove systems from a group.

```
usage: group_removesystems GROUP <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

group_restore

Restore a system group.

```
usage: group_backup INPUTDIR [NAME] ...
```

14.4.13 help

List all available spacecmd commands with the help function. Check for additional help on a specific function by calling for example: `user_create --help`.

```
Documented commands (type help <topic>):
========================================
activationkey_addchildchannels         org_trustdetails
activationkey_addconfigchannels        package_details
activationkey_addentitlements          package_listdependencies
activationkey_addgroups                package_listerrata
activationkey_addpackages              package_listinstalledsystems
activationkey_clone                    package_listorphans
activationkey_create                   package_remove
activationkey_delete                   package_removeorphans
activationkey_details                  package_search
activationkey_diff                     repo_addfilters
activationkey_disable                  repo_clearfilters
activationkey_disableconfigdeployment  repo_create
activationkey_enable                   repo_delete
activationkey_enableconfigdeployment   repo_details
activationkey_export                   repo_list
activationkey_import                   repo_listfilters
activationkey_list                     repo_removefilters
activationkey_listbasechannel          repo_rename
activationkey_listchildchannels        repo_setfilters
activationkey_listconfigchannels       repo_updatessl
activationkey_listentitlements         repo_updateurl
activationkey_listgroups               report_duplicates
activationkey_listpackages             report_errata
activationkey_listsystems              report_inactivesystems
activationkey_removechildchannels      report_ipaddresses
activationkey_removeconfigchannels     report_kernels
activationkey_removeentitlements       report_outofdatesystems
activationkey_removegroups             report_ungroupedsystems
activationkey_removepackages           scap_getxccdfscandetails
activationkey_setbasechannel           scap_getxccdfscanruleresults
activationkey_setconfigchannelorder    scap_listxccdfscans
activationkey_setcontactmethod         scap_schedulexccdfscan
activationkey_setdescription           schedule_cancel
activationkey_setuniversaldefault      schedule_details
activationkey_setusagelimit            schedule_getoutput
api                                    schedule_list
```

```
clear                                  schedule_listarchived
clear_caches                           schedule_listcompleted
configchannel_addfile                  schedule_listfailed
configchannel_backup                   schedule_listpending
configchannel_clone                    schedule_reschedule
configchannel_create                   snippet_create
configchannel_delete                   snippet_delete
configchannel_details                  snippet_details
configchannel_diff                     snippet_list
configchannel_export                   snippet_update
configchannel_filedetails              softwarechannel_adderrata
configchannel_forcedeploy              softwarechannel_adderratabydate
configchannel_import                   softwarechannel_addpackages
configchannel_list                     softwarechannel_addrepo
configchannel_listfiles                softwarechannel_clone
configchannel_listsystems              softwarechannel_clonetree
configchannel_removefiles              softwarechannel_create
configchannel_sync                     softwarechannel_delete
configchannel_updatefile               softwarechannel_details
configchannel_verifyfile               softwarechannel_diff
cryptokey_create                       softwarechannel_errata_diff
cryptokey_delete                       softwarechannel_errata_sync
cryptokey_details                      softwarechannel_getorgaccess
cryptokey_list                         softwarechannel_list
custominfo_createkey                   softwarechannel_listallpackages
custominfo_deletekey                   softwarechannel_listbasechannels
custominfo_details                     softwarechannel_listchildchannels
custominfo_listkeys                    softwarechannel_listerrata
custominfo_updatekey                   softwarechannel_listerratabydate
distribution_create                    softwarechannel_listlatestpackages
distribution_delete                    softwarechannel_listpackages
distribution_details                   softwarechannel_listrepos
distribution_list                      softwarechannel_listsyncschedule
distribution_rename                    softwarechannel_listsystems
distribution_update                    softwarechannel_mirrorpackages
errata_apply                           softwarechannel_regenerateneededcache
errata_delete                          softwarechannel_regenerateyumcache
errata_details                         softwarechannel_removeerrata
errata_findbycve                       softwarechannel_removepackages
errata_list                            softwarechannel_removerepo
errata_listaffectedsystems             softwarechannel_removesyncschedule
errata_listcves                        softwarechannel_setorgaccess
errata_publish                         softwarechannel_setsyncschedule
errata_search                          softwarechannel_sync
errata_summary                         softwarechannel_syncrepos
filepreservation_create                ssm_add
filepreservation_delete                ssm_clear
```

```
filepreservation_details             ssm_intersect
filepreservation_list                ssm_list
get_apiversion                       ssm_remove
get_certificateexpiration            system_addchildchannels
get_serverversion                    system_addconfigchannels
get_session                          system_addconfigfile
group_addsystems                     system_addcustomvalue
group_backup                         system_addentitlements
group_create                         system_addnote
group_delete                         system_applyerrata
group_details                        system_comparepackageprofile
group_list                           system_comparepackages
group_listsystems                    system_comparewithchannel
group_removesystems                  system_createpackageprofile
group_restore                        system_delete
help                                 system_deletecrashes
history                              system_deletenotes
kickstart_addactivationkeys          system_deletepackageprofile
kickstart_addchildchannels           system_deployconfigfiles
kickstart_addcryptokeys              system_details
kickstart_addfilepreservations       system_getcrashfiles
kickstart_addoption                  system_installpackage
kickstart_addpackages                system_list
kickstart_addscript                  system_listbasechannel
kickstart_addvariable                system_listchildchannels
kickstart_clone                      system_listconfigchannels
kickstart_create                     system_listconfigfiles
kickstart_delete                     system_listcrashedsystems
kickstart_details                    system_listcrashesbysystem
kickstart_diff                       system_listcustomvalues
kickstart_disableconfigmanagement    system_listentitlements
kickstart_disableremotecommands      system_listerrata
kickstart_enableconfigmanagement     system_listevents
kickstart_enablelogging              system_listhardware
kickstart_enableremotecommands       system_listinstalledpackages
kickstart_export                     system_listnotes
kickstart_getcontents                system_listpackageprofiles
kickstart_getsoftwaredetails         system_listupgrades
kickstart_getupdatetype              system_lock
kickstart_import                     system_reboot
kickstart_import_raw                 system_removechildchannels
kickstart_importjson                 system_removeconfigchannels
kickstart_list                       system_removecustomvalues
kickstart_listactivationkeys         system_removeentitlement
kickstart_listchildchannels          system_removepackage
kickstart_listcryptokeys             system_rename
kickstart_listcustomoptions          system_runscript
```

```
kickstart_listoptions                   system_schedulehardwarerefresh
kickstart_listpackages                  system_schedulepackagerefresh
kickstart_listscripts                   system_search
kickstart_listvariables                 system_setbasechannel
kickstart_removeactivationkeys          system_setconfigchannelorder
kickstart_removechildchannels           system_setcontactmethod
kickstart_removecryptokeys              system_show_packageversion
kickstart_removefilepreservations       system_syncpackages
kickstart_removeoptions                 system_unlock
kickstart_removepackages                system_updatecustomvalue
kickstart_removescript                  system_upgradepackage
kickstart_removevariables               toggle_confirmations
kickstart_rename                        user_adddefaultgroup
kickstart_setcustomoptions              user_addgroup
kickstart_setdistribution               user_addrole
kickstart_setlocale                     user_create
kickstart_setpartitions                 user_delete
kickstart_setselinux                    user_details
kickstartsetupdatetype                 user_disable
kickstart_updatevariable                user_enable
list_proxies                            user_list
login                                   user_listavailableroles
logout                                  user_removedefaultgroup
org_addtrust                            user_removegroup
org_create                              user_removerole
org_delete                              user_setemail
org_details                             user_setfirstname
org_list                                user_setlastname
org_listtrusts                          user_setpassword
org_listusers                           user_setprefix
org_removetrust                         whoami
org_rename                              whoamitalkingto


Miscellaneous help topics:
==========================
time  systems  ssm
```

14.4.14 history

List recent commands using the `history` command.

```
spacecmd {SSM:0}> history
   1  help
   2  api
   3  exit
   4  help
   5  time --help
   6  quit
   7  clear
spacecmd {SSM:0}>
```

14.4.15 kickstart_

The following spacecmd functions are available for use with kickstart.

kickstart_addactivationkeys

Add activation keys to a Kickstart profile.

```
usage: kickstart_addactivationkeys PROFILE <KEY ...>
```

kickstart_addchildchannels

Add a child channels to a Kickstart profile.

```
usage: kickstart_addchildchannels PROFILE <CHANNEL ...>
```

kickstart_addcryptokeys

Add cryptography keys to a Kickstart profile.

```
usage: kickstart_addcryptokeys PROFILE <KEY ...>
```

kickstart_addfilepreservations

Add file preservations to a Kickstart profile.

```
usage: kickstart_addfilepreservations PROFILE <FILELIST ...>
```

kickstart_addoption

Set an option for a Kickstart profile.

```
usage: kickstart_addoption PROFILE KEY [VALUE]
```

kickstart_addpackages

Add packages to a Kickstart profile.

```
usage: kickstart_addpackages PROFILE <PACKAGE ...>
```

kickstart_addscript

Add a script to a Kickstart profile.

```
usage: kickstart_addscript PROFILE [options]

options:
  -p PROFILE
  -e EXECUTION_TIME ['pre', 'post']
  -i INTERPRETER
  -f FILE
  -c execute in a chroot environment
  -t ENABLING_TEMPLATING
```

kickstart_addvariable

Add a variable to a Kickstart profile.

```
usage: kickstart_addvariable PROFILE KEY VALUE
```

kickstart_clone

Clone a Kickstart profile.

```
usage: kickstart_clone [options]

options:
  -n NAME
  -c CLONE_NAME
```

kickstart_create

Create a Kickstart profile.

```
usage: kickstart_create [options]

options:
  -n NAME
  -d DISTRIBUTION
  -p ROOT_PASSWORD
  -v VIRT_TYPE ['none', 'para_host', 'qemu', 'xenfv', 'xenpv']
```

kickstart_delete

Delete kickstart profile(s).

```
usage: kickstart_delete PROFILE
usage: kickstart_delete PROFILE1 PROFILE2
usage: kickstart_delete "PROF*"
```

kickstart_details

Show the details of a Kickstart profile.

```
usage: kickstart_details PROFILE
```

kickstart_diff

List differences between two kickstart files.

```
usage: kickstart_diff SOURCE_CHANNEL TARGET_CHANNEL
```

kickstart_disableconfigmanagement

Disable configuration management on a Kickstart profile.

```
usage: kickstart_disableconfigmanagement PROFILE
```

kickstart_disableremotecommands

Disable remote commands on a Kickstart profile.

```
usage: kickstart_disableremotecommands PROFILE
```

kickstart_enableconfigmanagement

Enable configuration management on a Kickstart profile.

```
usage: kickstart_enableconfigmanagement PROFILE
```

kickstart_enablelogging

Enable logging for a Kickstart profile.

```
usage: kickstart_enablelogging PROFILE
```

kickstart_enableremotecommands

Enable remote commands on a Kickstart profile.

```
usage: kickstart_enableremotecommands PROFILE
```

kickstart_export

Export kickstart profile(s) to json formatted file.

```
usage: kickstart_export <KSPROFILE>... [options]
options:
    -f outfile.json : specify an output filename, defaults to <KSPROFILE>.json
                      if exporting a single kickstart, profiles.json for multiple
                      kickstarts, or ks_all.json if no KSPROFILE specified
                      e.g (export ALL)

Note : KSPROFILE list is optional, default is to export ALL
```

kickstart_getcontents

Show the contents of a Kickstart profile as they would be presented to a client.

```
usage: kickstart_getcontents LABEL
```

kickstart_getsoftwaredetails

Gets kickstart profile software details.

```
usage: kickstart_getsoftwaredetails KS_LABEL
usage: kickstart_getsoftwaredetails KS_LABEL KS_LABEL2 ...
```

kickstart_getupdatetype

Get the update type for a kickstart profile(s).

```
usage: kickstart_getupdatetype PROFILE
usage: kickstart_getupdatetype PROFILE1 PROFILE2
usage: kickstart_getupdatetype "PROF*"
```

kickstart_import

Import a Kickstart profile from a file.

```
usage: kickstart_import [options]

options:
  -f FILE
  -n NAME
  -d DISTRIBUTION
  -v VIRT_TYPE ['none', 'para_host', 'qemu', 'xenfv', 'xenpv']
```

kickstart_import_raw

Import a raw Kickstart or autoyast profile from a file.

```
usage: kickstart_import_raw [options]

options:
  -f FILE
  -n NAME
  -d DISTRIBUTION
  -v VIRT_TYPE ['none', 'para_host', 'qemu', 'xenfv', 'xenpv']
```

kickstart_importjson

Import kickstart profile(s) from json file.

```
usage: kickstart_import <JSONFILES...>
```

kickstart_list

List the available Kickstart profiles.

```
usage: kickstart_list
```

kickstart_listactivationkeys

List the activation keys associated with a Kickstart profile.

```
usage: kickstart_listactivationkeys PROFILE
```

kickstart_listchildchannels

List the child channels of a Kickstart profile.

```
usage: kickstart_listchildchannels PROFILE
```

kickstart_listcryptokeys

List the crypto keys associated with a Kickstart profile.

```
usage: kickstart_listcryptokeys PROFILE
```

kickstart_listcustomoptions

List the custom options of a Kickstart profile.

```
usage: kickstart_listcustomoptions PROFILE
```

kickstart_listoptions

List the options of a Kickstart profile.

```
usage: kickstart_listoptions PROFILE
```

kickstart_listpackages

List the packages for a Kickstart profile.

```
usage: kickstart_listpackages PROFILE
```

kickstart_listscripts

List the scripts for a Kickstart profile.

```
usage: kickstart_listscripts PROFILE
```

kickstart_listvariables

List the variables of a Kickstart profile.

```
usage: kickstart_listvariables PROFILE
```

kickstart_removeactivationkeys

Remove activation keys from a Kickstart profile.

```
usage: kickstart_removeactivationkeys PROFILE <KEY ...>
```

kickstart_removechildchannels

Remove child channels from a Kickstart profile.

```
usage: kickstart_removechildchannels PROFILE <CHANNEL ...>
```

kickstart_removecryptokeys

Remove crypto keys from a Kickstart profile.

```
usage: kickstart_removecryptokeys PROFILE <KEY ...>
```

kickstart_removefilepreservations

Remove file preservations from a Kickstart profile.

```
usage: kickstart_removefilepreservations PROFILE <FILE ...>
```

kickstart_removeoptions

Remove options from a Kickstart profile.

```
usage: kickstart_removeoptions PROFILE <OPTION ...>
```

kickstart_removepackages

Remove packages from a Kickstart profile.

```
usage: kickstart_removepackages PROFILE <PACKAGE ...>
```

kickstart_removescript

Add a script to a Kickstart profile.

```
usage: kickstart_removescript PROFILE [ID]
```

kickstart_removevariables

Remove variables from a Kickstart profile.

```
usage: kickstart_removevariables PROFILE <KEY ...>
```

kickstart_rename

Rename a Kickstart profile

```
usage: kickstart_rename OLDNAME NEWNAME
```

kickstart_setcustomoptions

Set custom options for a Kickstart profile.

```
usage: kickstart_setcustomoptions PROFILE
```

kickstart_setdistribution

Set the distribution for a Kickstart profile.

```
usage: kickstart_setdistribution PROFILE DISTRIBUTION
```

kickstart_setlocale

Set the locale for a Kickstart profile.

```
usage: kickstart_setlocale PROFILE LOCALE
```

kickstart_setpartitions

Set the partitioning scheme for a Kickstart profile.

```
usage: kickstart_setpartitions PROFILE
```

kickstart_setselinux

Set the SELinux mode for a Kickstart profile.

```
usage: kickstart_setselinux PROFILE MODE
```

kickstartsetupdatetype

Set the update type for a kickstart profile(s).

```
usage: kickstartsetupdatetype [options] KS_LABEL


options:
    -u UPDATE_TYPE ['red_hat', 'all', 'none']
```

kickstart_updatevariable

Update a variable in a Kickstart profile.

```
usage: kickstart_updatevariable PROFILE KEY VALUE
```

14.4.16 list_proxies

The following spacecmd function is available for listing proxies.

list_proxies

List the proxies within the user's organization.

```
usage: list_proxies
```

14.4.17 login

Connect as a specific user to the SUSE manager server.

```
# spacecmd -- login <USERNAME>
```

14.4.18 logout

Logout from server as the current user.

```
# spacecmd -- logout
```

14.4.19 org_

The following spacecmd functions are available for use with organizations.

org_addtrust

Add a trust between two organizations

```
usage: org_addtrust YOUR_ORG ORG_TO_TRUST
```

org_create

Create an organization.

```
usage: org_create [options]

options:
  -n ORG_NAME
  -u USERNAME
  -P PREFIX (Dr., Mr., Miss, Mrs., Ms.)
  -f FIRST_NAME
  -l LAST_NAME
  -e EMAIL
  -p PASSWORD
  --pam enable PAM authentication
```

org_delete

Delete an organization.

```
usage: org_delete NAME
```

org_details

Show the details of an organization.

```
usage: org_details NAME
```

org_list

List all organizations.

```
usage: org_list
```

org_listtrusts

List an organization’s trusts.

```
org_listtrusts NAME
```

org_listusers

List an organization’s users.

```
org_listusers NAME
```

org_removetrust

Remove a trust between two organizations.

```
usage: org_removetrust YOUR_ORG TRUSTED_ORG
```

org_rename

Rename an organization.

```
usage: org_rename OLDNAME NEWNAME
```

org_trustdetails

Show the details of an organizational trust.

```
usage: org_trustdetails TRUSTED_ORG
```

14.4.20 package_

The following spacecmd functions are available for working with packages.

package_details

Show the details of a software package.

```
usage: package_details PACKAGE ...
```

package_listdependencies

List the dependencies for a package.

```
usage: package_listdependencies PACKAGE
```

package_listerrata

List the errata that provide this package.

```
usage: package_listerrata PACKAGE ...
```

package_listinstalledsystems

List the systems with a package installed.

```
usage: package_listinstalledsystems PACKAGE ...
```

package_listorphans

List packages that are not in a channel.

```
usage: package_listorphans
```

package_remove

Remove a package from SUSE Manager/Satellite

```
usage: package_remove PACKAGE ...
```

package_removeorphans

Remove packages that are not in a channel.

```
usage: package_removeorphans
```

package_search

Find packages that meet the given criteria.

```
usage: package_search NAME|QUERY

Example: package_search kernel

Advanced Search:
Available Fields: name, epoch, version, release, arch, description, summary
Example: name:kernel AND version:2.6.18 AND -description:devel
```

14.4.21 repo_

The following spacecmd functions are available for working with repositories.

repo_addfilters

Add filters for a user repository.

```
usage: repo_addfilters repo <filter ...>
```

repo_clearfilters

Clears the filters for a user repository.

```
usage: repo_clearfilters repo
```

repo_create

Create a user repository.

```
usage: repo_create <options>

options:
  -n, --name   name of repository
  -u, --url    url of repository

  --ca         SSL CA certificate (not required)
  --cert       SSL Client certificate (not required)
  --key        SSL Client key (not required)
```

repo_delete

Delete a user repository.

```
usage: repo_delete <repo ...>
```

repo_details

Show the details of a user repository.

```
usage: repo_details <repo ...>
```

repo_list

List all available user repository.

```
usage: repo_list
```

repo_listfilters

Show the filters for a user repository.

```
usage: repo_listfilters repo
```

repo_removefilters

Remove filters from a user repository.

```
usage: repo_removefilters repo <filter ...>
```

repo_rename

Rename a user repository.

```
usage: repo_rename OLDNAME NEWNAME
```

repo_setfilters

Set the filters for a user repo.

```
usage: repo_setfilters repo <filter ...>
```

repo_updatessl

Change the SSL certificates of a user repository.

```
usage: repo_updatessl <options>
options:
  --ca         SSL CA certificate (not required)
  --cert       SSL Client certificate (not required)
  --key        SSL Client key (not required)
```

repo_updateurl

Change the URL of a user repository.

```
usage: repo_updateurl <repo> <url>
```

14.4.22 report_

The following spacecmd functions are available for working with reports.

report_duplicates

List duplicate system profiles.

```
usage: report_duplicates
```

report_errata

List all errata and how many systems they affect.

```
usage: report_errata [ERRATA|search:XXX ...]
```

report_inactivesystems

List all inactive systems.

```
usage: report_inactivesystems [DAYS]
```

report_ipaddresses

List the hostname and IP of each system.

```
usage: report_network [<SYSTEMS>]

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

report_kernels

List the running kernel of each system.

```
usage: report_kernels [<SYSTEMS>]

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

report_outofdatesystems

List all out-of-date systems.

```
usage: report_outofdatesystems
```

report_ungroupedsystems

List all ungrouped systems.

```
usage: report_ungroupedsystems
```

14.4.23 scap_

The following spacecmd functions are available for working with OpenSCAP.

scap_getxccdfscandetails

Get details of given OpenSCAP XCCDF scan.

```
usage: scap_getxccdfscandetails <XID>
```

scap_getxccdfscanruleresults

Return a full list of RuleResults for given OpenSCAP XCCDF scan.

```
usage: scap_getxccdfscanruleresults <XID>
```

scap_listxccdfscans

Return a list of finished OpenSCAP scans for given systems.

```
usage: scap_listxccdfscans <SYSTEMS>
```

scap_schedulexccdfscan

Schedule Scap XCCDF scan.

```
usage: scap_schedulexccdfscan PATH_TO_XCCDF_FILE XCCDF_OPTIONS SYSTEMS

Example:
> scap_schedulexccdfscan '/usr/share/openscap/scap-security-xccdf.xml' 'profile Web-
Default' \
system-scap.example.com
```

14.4.24 schedule_

The following spacecmd functions are available for working with scheduling.

schedule_cancel

Cancel a scheduled action.

```
usage: schedule_cancel ID|* ...
```

schedule_details

Show the details of a scheduled action.

```
usage: schedule_details ID
```

schedule_getoutput

Show the output from an action.

```
usage: schedule_getoutput ID
```

schedule_list

List all actions.

```
usage: schedule_list [BEGINDATE] [ENDDATE]

Dates can be any of the following:
Explicit Dates:
Dates can be expressed as explicit date strings in the YYYYMMDD[HHMM]
format.  The year, month and day are required, while the hours and
minutes are not; the hours and minutes will default to 0000 if no
values are provided.

Deltas:
Dates can be expressed as delta values.  For example, '2h' would
```

```
mean 2 hours in the future.  You can also use negative values to
express times in the past (e.g., -7d would be one week ago).

Units:
s -> seconds
m -> minutes
h -> hours
d -> days
```

schedule_listarchived

List archived actions.

```
usage: schedule_listarchived [BEGINDATE] [ENDDATE]

Dates can be any of the following:
Explicit Dates:
Dates can be expressed as explicit date strings in the YYYYMMDD[HHMM]
format.  The year, month and day are required, while the hours and
minutes are not; the hours and minutes will default to 0000 if no
values are provided.

Deltas:
Dates can be expressed as delta values.  For example, '2h' would
mean 2 hours in the future.  You can also use negative values to
express times in the past (e.g., -7d would be one week ago).

Units:
s -> seconds
m -> minutes
h -> hours
d -> days
```

schedule_listcompleted

List completed actions.

```
Dates can be any of the following:
Explicit Dates:
Dates can be expressed as explicit date strings in the YYYYMMDD[HHMM]
format.  The year, month and day are required, while the hours and
minutes are not; the hours and minutes will default to 0000 if no
values are provided.

Deltas:
Dates can be expressed as delta values.  For example, '2h' would
mean 2 hours in the future.  You can also use negative values to
express times in the past (e.g., -7d would be one week ago).
```

```
Units:
s -> seconds
m -> minutes
h -> hours
d -> days
```

schedule_listfailed

List failed actions.

```
usage: schedule_listfailed [BEGINDATE] [ENDDATE]

Dates can be any of the following:
Explicit Dates:
Dates can be expressed as explicit date strings in the YYYYMMDD[HHMM]
format.  The year, month and day are required, while the hours and
minutes are not; the hours and minutes will default to 0000 if no
values are provided.

Deltas:
Dates can be expressed as delta values.  For example, '2h' would
mean 2 hours in the future.  You can also use negative values to
express times in the past (e.g., -7d would be one week ago).

Units:
s -> seconds
m -> minutes
h -> hours
d -> days
```

schedule_listpending

List pending actions.

```
usage: schedule_listpending [BEGINDATE] [ENDDATE]

Dates can be any of the following:
Explicit Dates:
Dates can be expressed as explicit date strings in the YYYYMMDD[HHMM]
format.  The year, month and day are required, while the hours and
minutes are not; the hours and minutes will default to 0000 if no
values are provided.

Deltas:
Dates can be expressed as delta values.  For example, '2h' would
mean 2 hours in the future.  You can also use negative values to
express times in the past (e.g., -7d would be one week ago).
```

```

Units:
s -> seconds
m -> minutes
h -> hours
d -> days
```

schedule_reschedule

Reschedule failed actions.

```
usage: schedule_reschedule ID|* ...
```

14.4.25 snippet_

The following spacecmd functions are available for working with Kickstart snippets.

snippet_create

Create a Kickstart snippet

```
usage: snippet_create [options]

options:
  -n NAME
  -f FILE
```

snippet_delete

Delete a Kickstart snippet.

```
usage: snippet_removefile NAME
```

snippet_details

Show the contents of a snippet.

```
usage: snippet_details SNIPPET ...
```

snippet_list

List the available Kickstart snippets.

```
usage: snippet_list
```

snippet_update

Update a Kickstart snippet.

```
usage: snippet_update NAME
```

14.4.26 softwarechannel_

The following spacecmd functions are available for working with software channels.

softwarechannel_adderrata

Add patches from one channel into another channel.

```
usage: softwarechannel_adderrata SOURCE DEST <ERRATA|search:XXX ...>
Options:
    -q/--quick : Don't display list of packages (slightly faster)
    -s/--skip :  Skip errata which appear to exist already in DEST
```

softwarechannel_adderratabydate

Add errata from one channel into another channel based on a date range.

```
usage: softwarechannel_adderratabydate [options] SOURCE DEST BEGINDATE ENDDATE
Date format : YYYYMMDD
Options:
        -p/--publish : Publish errata to the channel (don't clone)
```

softwarechannel_addpackages

Add packages to a software channel.

```
usage: softwarechannel_addpackages CHANNEL <PACKAGE ...>
```

softwarechannel_addrepo

Add a repo to a software channel.

```
usage: softwarechannel_addrepo CHANNEL REPO
```

softwarechannel_clone

Clone a software channel.

```
usage: softwarechannel_clone [options]

options:
  -s SOURCE_CHANNEL
  -n NAME
  -l LABEL
  -p PARENT_CHANNEL
  --gpg-copy/-g (copy SOURCE_CHANNEL GPG details)
  --gpg-url GPG_URL
  --gpg-id GPG_ID
  --gpg-fingerprint GPG_FINGERPRINT
  -o do not clone any patches
  --regex/-x "s/foo/bar" : Optional regex replacement,
        replaces foo with bar in the clone name and label
```

softwarechannel_clonetree

Clone a software channel and its child channels.

```
usage: softwarechannel_clonetree [options]A
             e.g    softwarechannel_clonetree foobasechannel -p "my_"
                    softwarechannel_clonetree foobasechannel -x "s/foo/bar"
                    softwarechannel_clonetree foobasechannel -x "s/^/my_"

options:
  -s/--source-channel SOURCE_CHANNEL
  -p/--prefix PREFIX (is prepended to the label and name of all channels)
  --gpg-copy/-g (copy GPG details for correspondoing source channel))
  --gpg-url GPG_URL (applied to all channels)
  --gpg-id GPG_ID (applied to all channels)
  --gpg-fingerprint GPG_FINGERPRINT (applied to all channels)
  -o do not clone any errata
  --regex/-x "s/foo/bar" : Optional regex replacement,
        replaces foo with bar in the clone name, label and description
```

softwarechannel_create

Create a software channel.

```
usage: softwarechannel_create [options]

options:
  -n NAME
  -l LABEL
  -p PARENT_CHANNEL
  -a ARCHITECTURE ['ia32', 'ia64', 'x86_64', 'ppc',
                  'i386-sun-solaris', 'sparc-sun-solaris']
  -c CHECKSUM ['sha1', 'sha256', 'sha384', 'sha512']
  -u GPG_URL
  -i GPG_ID
  -f GPG_FINGERPRINT
```

softwarechannel_delete

Delete a software channel.

```
usage: softwarechannel_delete <CHANNEL ...>
```

softwarechannel_details

Show the details of a software channel.

```
usage: softwarechannel_details <CHANNEL ...>
```

softwarechannel_diff

Check the difference between software channels.

```
usage: softwarechannel_diff SOURCE_CHANNEL TARGET_CHANNEL
```

softwarechannel_errata_diff

Check the difference between software channel files.

```
usage: softwarechannel_errata_diff SOURCE_CHANNEL TARGET_CHANNEL
```

softwarechannel_errata_sync

Sync errata of two software channels.

```
usage: softwarechannel_errata_sync SOURCE_CHANNEL TARGET_CHANNEL
```

softwarechannel_getorgaccess

Get the org-access for the software channel.

```
usage : softwarechannel_getorgaccess : get org access for all channels
usage : softwarechannel_getorgaccess <channel_label(s)> : get org access for
 specific channel(s)
```

softwarechannel_list

List all available software channels.

```
usage: softwarechannel_list [options]'
options:
  -v verbose (display label and summary)
  -t tree view (pretty-print child-channels)
```

softwarechannel_listallpackages

List all packages in a channel.

```
usage: softwarechannel_listallpackages CHANNEL
```

softwarechannel_listbasechannels

List all base software channels.

```
usage: softwarechannel_listbasechannels [options]
options:
  -v verbose (display label and summary)
```

softwarechannel_listchildchannels

List child software channels.

```
usage:
softwarechannel_listchildchannels [options]
softwarechannel_listchildchannels : List all child channels
softwarechannel_listchildchannels CHANNEL : List children for a specific base
 channel
options:
 -v verbose (display label and summary)
```

softwarechannel_listerrata

List the errata associated with a software channel.

```
usage: softwarechannel_listerrata <CHANNEL ...> [from=yyyymmdd [to=yyyymmdd]]
```

softwarechannel_listerratabydate

List errata from channelbased on a date range.

```
usage: softwarechannel_listerratabydate CHANNEL BEGINDATE ENDDATE
Date format : YYYYMMDD
```

softwarechannel_listlatestpackages

List the newest version of all packages in a channel.

```
usage: softwarechannel_listlatestpackages CHANNEL
```

softwarechannel_listpackages

List the most recent packages available from a software channel.

```
usage: softwarechannel_listpackages CHANNEL
```

softwarechannel_listrepos

List the repos for a software channel.

```
usage: softwarechannel_listrepos CHANNEL
```

softwarechannel_listsyncschedule

List sync schedules for all software channels.

```
usage: softwarechannel_listsyncschedule : List all channels
```

softwarechannel_listsystems

List all systems subscribed to a software channel.

```
usage: softwarechannel_listsystems CHANNEL
```

softwarechannel_mirrorpackages

Download packages of a given channel.

```
usage: softwarechannel_mirrorpackages CHANNEL
Options:
    -l/--latest : Only mirror latest package version
```

softwarechannel_regenerateneededcache

Regenerate the needed errata and package cache for all systems.

```
usage: softwarechannel_regenerateneededcache
```

softwarechannel_regenerateyumcache

Regenerate the YUM cache for a software channel.

```
usage: softwarechannel_regenerateyumcache <CHANNEL ...>
```

softwarechannel_removeerrata

Remove patches from a software channel.

```
usage: softwarechannel_removeerrata CHANNEL <ERRATA:search:XXX ...>
```

softwarechannel_removepackages

Remove packages from a software channel.

```
usage: softwarechannel_removepackages CHANNEL <PACKAGE ...>
```

softwarechannel_removerepo

Remove a repo from a software channel.

```
usage: softwarechannel_removerepo CHANNEL REPO
```

softwarechannel_removesyncschedule

Removes the repo sync schedule for a software channel.

```
usage: softwarechannel_setsyncschedule <CHANNEL>
```

softwarechannel_setorgaccess

Set the org-access for the software channel.

```
usage : softwarechannel_setorgaccess <channel_label> [options]
-d,--disable : disable org access (private, no org sharing)
-e,--enable : enable org access (public access to all trusted orgs)
```

softwarechannel_setsyncschedule

Sets the repo sync schedule for a software channel.

```
usage: softwarechannel_setsyncschedule <CHANNEL> <SCHEDULE>

The schedule is specified in Quartz CronTrigger format without enclosing quotes.
For example, to set a schedule of every day at 1am, <SCHEDULE> would be 0 0 1 * * ?
```

softwarechannel_sync

Sync the packages of two software channels.

```
usage: softwarechannel_sync SOURCE_CHANNEL TARGET_CHANNEL
```

softwarechannel_syncrepos

Sync users repos for a software channel.

```
usage: softwarechannel_syncrepos <CHANNEL ...>
```

14.4.27 ssm_

The following spacecmd functions are available for use with System Set Manager.

ssm_add

Add systems to the SSM.

```
usage: ssm_add <SYSTEMS>

see 'help ssm' for more details

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
```

```
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

ssm_clear

Remove all systems from the SSM.

```
usage: ssm_clear
```

ssm_intersect

Replace the current SSM with the intersection of the current list of systems and the list of systems passed as arguments.

```
usage: ssm_intersect <SYSTEMS>

see 'help ssm' for more details

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNE
```

ssm_list

List the systems currently in the SSM.

```
usage: ssm_list

see 'help ssm' for more details
```

ssm_remove

Remove systems from the SSM.

```
usage: ssm_remove <SYSTEMS>

see 'help ssm' for more details

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

14.4.28 system_

The following spacecmd functions are available for use with systems.

system_addchildchannels

Add child channels to a system.

```
usage: system_addchildchannels <SYSTEMS> <CHANNEL ...>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_addconfigchannels

Add config channels to a system.

```
usage: system_addconfigchannels <SYSTEMS> <CHANNEL ...> [options]

options:
  -t add channels to the top of the list
  -b add channels to the bottom of the list

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_addconfigfile

Create a configuration file.

```
Note this is only for system sandbox or locally-managed files
Centrally managed files should be created via configchannel_addfile
usage: system_addconfigfile [SYSTEM] [options]

options:
  -S/--sandbox : list only system-sandbox files
  -L/--local   : list only locally managed files
  -p PATH
  -r REVISION
  -o OWNER [default: root]
  -g GROUP [default: root]
  -m MODE [defualt: 0644]
  -x SELINUX_CONTEXT
  -d path is a directory
  -s path is a symlink
  -b path is a binary (or other file which needs base64 encoding)
  -t SYMLINK_TARGET
  -f local path to file contents

  Note re binary/base64: Some text files, notably those containing trailing
  newlines, those containing ASCII escape characters (or other charaters not
  allowed in XML) need to be sent as binary (-b).  Some effort is made to auto-
  detect files which require this, but you may need to explicitly specify.
```

system_addcustomvalue

Set a custom value for a system.

```
usage: system_addcustomvalue KEY VALUE <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_addentitlements

Add entitlements to a system.

```
usage: system_addentitlements <SYSTEMS> ENTITLEMENT

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_addnote

Set a note for a system.

```
usage: system_addnote <SYSTEM> [options]

options:
  -s SUBJECT
  -b BODY
```

```

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_applyerrata

Apply errata to a system.

```
usage: system_applyerrata <SYSTEMS> [ERRATA|search:XXX ...]

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_comparepackageprofile

Compare a system against a package profile.

```
usage: system_comparepackageprofile <SYSTEMS> PROFILE

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_comparepackages

Compare the packages between two systems.

```
usage: system_comparepackages SOME_SYSTEM ANOTHER_SYSTEM
```

system_comparewithchannel

Compare the installed packages on a system with those in the channels it is registered to, or optionally some other channel.

```
usage: system_comparewithchannel <SYSTEMS> [options]
options:
         -c/--channel : Specific channel to compare against,
                        default is those subscribed to, including
                        child channels

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_createpackageprofile

Create a package profile.

```
usage: system_createpackageprofile SYSTEM [options]

options:
  -n NAME
  -d DESCRIPTION
```

system_delete

Delete a system profile.

```
usage: system_delete <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_deletecrashes

Delete crashes reported by spacewalk-abrt.

```
usage: Delete all crashes for all systems    : system_deletecrashes [--verbose]
usage: Delete all crashes for a single system: system_deletecrashes -i sys_id [--
verbose]
usage: Delete a single crash record          : system_deletecrashes -c crash_id [--
verbose]
```

system_deletenotes

Delete notes from a system.

```
usage: system_deletenotes <SYSTEM> <ID|*>
```

```
<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_deletepackageprofile

Delete a package profile.

```
usage: system_deletepackageprofile PROFILE
```

system_deployconfigfiles

Deploy all configuration files for a system.

```
usage: system_deployconfigfiles <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_details

Show the details of a system profile.

```
usage: system_details <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_getcrashfiles

Download all files for a crash record.

```
usage: system_getcrashfiles -c crash_id [--verbose]
usage: system_getcrashfiles -c crash_id [--dest_folder=/tmp/crash_files] [--verbose]
```

system_installpackage

Install a package on a system.

```
usage: system_installpackage <SYSTEMS> <PACKAGE ...>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_list

List all system profiles.

```
usage: system_list
```

system_listbasechannel

List the base channel for a system.

```
usage: system_listbasechannel <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_listchildchannels

List the child channels for a system.

```
usage: system_listchildchannels <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_listconfigchannels

List the config channels of a system.

```
usage: system_listconfigchannels <SYSTEMS>

<SYSTEMS> can be any of the following:
```

```
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_listconfigfiles

List the managed config files of a system.

```
usage: system_listconfigfiles <SYSTEMS>'
options:
  -s/--sandbox : list only system-sandbox files
  -l/--local   : list only locally managed files
  -c/--central : list only centrally managed files
  -q/--quiet   : quiet mode (omits the header)

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_listcrashedsystems

List all systems that have experienced a crash and reported by spacewalk-abrt.

```
usage: system_listcrashedsystems
```

system_listcrashesbysystem

List all reported crashes for a system.

```
usage: system_listcrashesbysystem -i sys_id
```

system_listcustomvalues

List the custom values for a system.

```
usage: system_listcustomvalues <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_listentitlements

List the entitlements for a system.

```
usage: system_listentitlements <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_listerrata

List available errata for a system.

```
usage: system_listerrata <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_listevents

List the event history for a system.

```
usage: system_listevents <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_listhardware

List the hardware details of a system.

```
usage: system_listhardware <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
```

```
group:GROUP
channel:CHANNEL
```

system_listinstalledpackages

List the installed packages on a system.

```
usage: system_listinstalledpackages <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_listnotes

List the available notes for a system.

```
usage: system_listnotes <SYSTEM>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_listpackageprofiles

List all package profiles.

```
usage: system_listpackageprofiles
```

system_listupgrades

List the available upgrades for a system.

```
usage: system_listupgrades <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_lock

Lock a system.

```
usage: system_lock <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_reboot

Reboot a system.

```
usage: system_reboot <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_removechildchannels

Remove child channels from a system.

```
usage: system_removechildchannels <SYSTEMS> <CHANNEL ...>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_removeconfigchannels

Remove config channels from a system.

```
usage: system_removeconfigchannels <SYSTEMS> <CHANNEL ...>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
```

```
group:GROUP
channel:CHANNEL
```

system_removecustomvalues

Remove a custom value for a system.

```
usage: system_removecustomvalues <SYSTEMS> <KEY ...>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_removeentitlement

Remove an entitlement from a system.

```
usage: system_removeentitlement <SYSTEMS> ENTITLEMENT

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_removepackage

Remove a package from a system.

```
usage: system_removepackage <SYSTEMS> <PACKAGE ...>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_rename

Rename a system profile.

```
usage: system_rename OLDNAME NEWNAME
```

system_runscript

Schedule a script to run on the list of systems provided.

```
usage: system_runscript <SYSTEMS> [options]

options:
  -u USER
  -g GROUP
  -t TIMEOUT
  -s START_TIME
  -l LABEL
  -f FILE

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL



Dates can be any of the following:
Explicit Dates:
Dates can be expressed as explicit date strings in the YYYYMMDD[HHMM]
format.  The year, month and day are required, while the hours and
minutes are not; the hours and minutes will default to 0000 if no
values are provided.

Deltas:
Dates can be expressed as delta values.  For example, '2h' would
mean 2 hours in the future.  You can also use negative values to
express times in the past (e.g., -7d would be one week ago).

Units:
s -> seconds
m -> minutes
h -> hours
d -> days
```

system_schedulehardwarerefresh

Schedule a hardware refresh for a system.

```
usage: system_schedulehardwarerefresh <SYSTEMS>

<SYSTEMS> can be any of the following:
name
```

```
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_schedulepackagerefresh

Schedule a software package refresh for a system.

```
usage: system_schedulepackagerefresh <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_search

List systems that match the given criteria.

```
usage: system_search QUERY

Available Fields:
id
name
ip
hostname
device
vendor
driver
uuid

Examples:
> system_search device:vmware
> system_search ip:192.168.82
```

system_setbasechannel

Set a system’s base software channel.

```
usage: system_setbasechannel <SYSTEMS> CHANNEL

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
```

```
group:GROUP
channel:CHANNEL
```

system_setconfigchannelorder

Set the ranked order of configuration channels.

```
usage: system_setconfigchannelorder <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_setcontactmethod

Set the contact method for given system(s).

```
Available contact methods: ['default', 'ssh-push', 'ssh-push-tunnel']
usage: system_setcontactmethod <SYSTEMS> <CONTACT_METHOD>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_show_packageversion

Shows version of installed package on given system(s).

```
usage: system_show_packageversion <SYSTEM> <PACKAGE>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_syncpackages

Sync packages between two systems.

```
usage: system_syncpackages SOURCE TARGET
```

system_unlock

Unlock a system.

```
usage: system_unlock <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_updatecustomvalue

Update a custom value for a system.

```
usage: system_updatecustomvalue KEY VALUE <SYSTEMS>

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

system_upgradepackage

Upgrade a package on a system.

```
usage: system_upgradepackage <SYSTEMS> <PACKAGE ...>|*

<SYSTEMS> can be any of the following:
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

14.4.29 toggle_

toggle_confirmations

Toggle confirmation messages on/off.

```
usage: toggle_confirmations
```

14.4.30 user_

user_adddefaultgroup

Add a default group to an user account.

```
usage: user_adddefaultgroup USER <GROUP ...>
```

user_addgroup

Add a group to an user account.

```
usage: user_addgroup USER <GROUP ...>
```

user_addrole

Add a role to an user account.

```
usage: user_addrole USER ROLE
```

user_create

Create an user.

```
usage: user_create [options]

options:
  -u USERNAME
  -f FIRST_NAME
  -l LAST_NAME
  -e EMAIL
  -p PASSWORD
  --pam enable PAM authentication
```

user_delete

Delete an user.

```
usage: user_delete NAME
```

user_details

Show the details of an user.

```
usage: user_details USER ...
```

user_disable

Disable an user account.

```
usage: user_disable NAME
```

user_enable

Enable an user account.

```
usage: user_enable NAME
```

user_list

List all users.

```
usage: user_list
```

user_listavailableroles

List all available roles for users.

```
usage: user_listavailableroles
```

user_removedefaultgroup

Remove a default group from an user account.

```
usage: user_removedefaultgroup USER <GROUP ...>
```

user_removegroup

Remove a group to an user account.

```
usage: user_removegroup USER <GROUP ...>
```

user_removerole

Remove a role from an user account.

```
usage: user_removerole USER ROLE
```

user_setemail

Set an user accounts email field.

```
usage: user_setemail USER EMAIL
```

user_setfirstname

Set an user accounts first name field.

```
usage: user_setfirstname USER FIRST_NAME
```

user_setlastname

Set an user accounts last name field.

```
usage: user_setlastname USER LAST_NAME
```

user_setpassword

Set an user accounts name prefix field.

```
usage: user_setpassword USER PASSWORD
```

user_setprefix

Set an user accounts name prefix field.

```
usage: user_setprefix USER PREFIX
```

14.4.31 whoami

The following command is available for returning the currently logged spacecmd username.

whoami

Print the currently logged spacecmd user.

```
spacecmd {SSM:0}> whoami
admin
```

14.4.32 whoamitalkingto

The following spacecmd function is available for returning the server hostname.

whoamitalkingto

Return the server hostname that spacecmd is connected with.

```
spacecmd {SSM:0}> whoamitalkingto
MGR_SERVER_HOSTNAME
```

14.4.33 Miscellaneous Help Topics

The following help topics are printed with all functions requiring the relevant information.

time

Dates can be any of the following:

```
Explicit Dates:
Dates can be expressed as explicit date strings in the YYYYMMDD[HHMM]
format.  The year, month and day are required, while the hours and
minutes are not; the hours and minutes will default to 0000 if no
values are provided.

Deltas:
Dates can be expressed as delta values.  For example, '2h' would
mean 2 hours in the future.  You can also use negative values to
express times in the past (e.g., -7d would be one week ago).

Units:
s -> seconds
m -> minutes
h -> hours
d -> days
```

systems

<SYSTEMS> can be any of the following:

```
name
ssm (see 'help ssm')
search:QUERY (see 'help system_search')
group:GROUP
channel:CHANNEL
```

ssm

The System Set Manager (SSM) is a group of systems that you
can perform tasks on as a group.

```
Adding Systems:
> ssm_add group:rhel5-x86_64
> ssm_add channel:rhel-x86_64-server-5
> ssm_add search:device:vmware
> ssm_add host.example.com

Intersections:
> ssm_add group:rhel5-x86_64
> ssm_intersect group:web-servers

Using the SSM:
> system_installpackage ssm zsh
> system_runscript ssm
```

A Ports

A.1 Uyuni Server

Some ports are only relevant if you actually run the related service on the Uyuni server.

PORTS TO OPEN ON UYUNISERVER

67

Inbound / TCP/UDP / DHCP

Required when Uyuni is configured as a DHCP server for systems requesting IP addresses.

69

Inbound / TCP/UDP / TFTP

Used when Uyuni is configured as a PXE server and allows installation and re-installation of PXE-boot enabled systems.

80

Inbound / TCP / HTTP

Client and proxy server requests travel via HTTP or HTTPS.

80

Outbound / TCP / HTTP

Used to contact SUSE Customer Center/Novell Customer Center.

443

Inbound / TCP / HTTPS

All Web UI, client, and proxy server requests travel via HTTP or HTTPS.

443

Outbound / TCP / HTTPS

Uyuni uses this port to reach SUSE Customer Center (unless running in a disconnected mode with RMT or SMT-as described in).

5222

Inbound / TCP / osad

When you wish to push actions to clients this port is required by the `osad` daemon running on your client systems.

5269

Inbound/Outbound / TCP / jabberd

Needed if you push actions to or via a SUSE Manager Proxy.

4505

Inbound / TCP / salt

Required by the Salt-master to accept communication requests via TCP from minions. The connection is initiated by the minion and remains open to allow the master to send commands. This port uses a publish/subscribe topology; the minion subscribes to notifications from the master.

4506

Inbound / TCP / salt

Required by the Salt-master to accept communication requests via TCP from minions. The connection is initiated by the minion and is open only when needed. Usually, minions will open this port when they have to report results to the master, such as when a command received on port 4505 has finished. This port uses a request/response topology; the minion sends requests to the master.

25151

TCP

For cobbler.

INTERNALLY USED PORTS ON UYUNI SERVER

2828

Internal /

Satellite-search API, used by the RHN application in Tomcat and Taskomatic.

2829

Internal /

Taskomatic API, used by the RHN application in Tomcat.

6868

Internal

Auditlog-keeper to database.

6888

Internal

Auditlog-keeper API, used by the RHN application in Tomcat.

8005

Internal

Tomcat shutdown port.

8009

Internal

Tomcat to Apache HTTPD (AJP).

8080

Internal

Tomcat to Apache HTTPD (HTTP).

9080

Internal

Salt-API, used by the RHN application in Tomcat and Taskomatic.

32000

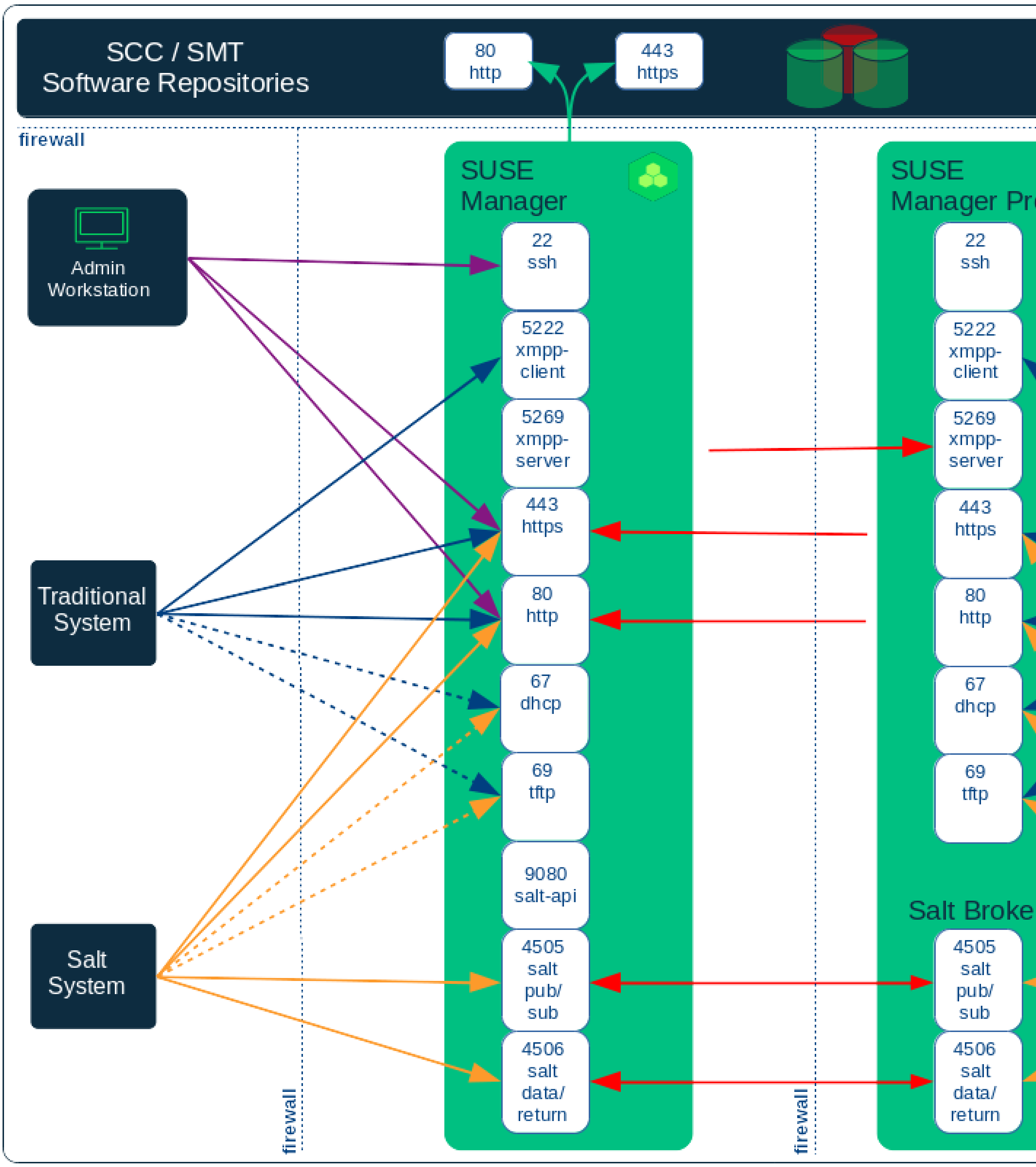
Internal / TCP

Port for a TCP connection to the Java Virtual Machine (JVM) that runs Taskomatic and the search (satellite-search).

Note: Ephemeral Ports

Anything from port 32768 on (more exactly, what you can see with `cat /proc/sys/net/ipv4/ip_local_port_range`) is an ephemeral port, typically used as the receiving end of a TCP connection. So if process A opens a TCP connection to process B (for example, port 22), then A chooses an arbitrary source TCP port to match with destination port 22.

This image is a graphical representation of the ports used in Uyuni:



Port 80 (http) is not used to serve the Web UI, and is closed in most installations. Port 80 is used temporarily for some bootstrap repositories and automated installations.

A.2 Uyuni Proxy Server

PORTS TO OPEN ON UYUNIPROXY SERVER

22

Inbound /

Required when using ssh-push or ssh-push-tunnel contact methods. Check-in on clients connected to a Uyuni Proxy will be initiated on the Uyuni Server and “hop through” through to clients.

80

Outbound /

Used to reach Uyuni.

5222

Inbound / TCP

For push actions and connections issued by `osad` running on the client systems.

5269

Inbound/Outbound / TCP

For push actions with the server.

A.3 Uyuni Client

PORTS TO OPEN ON UYUNI CLIENT

22

Inbound / SSH

Required when using ssh-push or ssh-push-tunnel contact methods.

80

Outbound

To reach the Uyuni server or SUSE Manager Proxy server.

5222

Outbound / TCP

For push actions with the server or proxy server.

B Minimalist AutoYaST Profile for Automated Installations and Useful Enhancements

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

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

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

The complete AutoYaST file:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns"
         xmlns:config="http://www.suse.com/1.0/configns">
 <general>
  <mode>
   <confirm config:type="boolean">false</confirm>
  </mode>
 </general>
 <networking>
  <keep_install_network config:type="boolean">true</keep_install_network>
 </networking>
 <software>
  <install_recommended config:type="boolean">true</install_recommended>
   <patterns config:type="list">
    <pattern>base</pattern>
   </patterns>
 </software>
 <users config:type="list">
  <user>
   <encrypted config:type="boolean">false</encrypted>
   <fullname>root</fullname>
   <gid>0</gid>
   <home>/root</home>
   <password_settings>
```

```
    <expire></expire>
    <flag></flag>
    <inact></inact>
    <max></max>
    <min></min>
    <warn></warn>
   </password_settings>
   <shell>/bin/bash</shell>
   <uid>0</uid>
   <username>root</username>
   <user_password>linux</user_password>
  </user>
 </users>
 <scripts>
  <init-scripts config:type="list">
   <script>
    <interpreter>shell</interpreter>
    <location>http://192.168.1.1/pub/bootstrap/my_bootstrap.sh</location>
   </script>
  </init-scripts>
 </scripts>
</profile>
```

Use this enhancement fragment to add child channels:

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

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

Important: Add the Updates Channel

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

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

C GNU Licenses

This appendix contains the GNU Free Documentation License version 1.2.

15 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.

9. TERMINATION

10. FUTURE REVISIONS OF THIS LICENSE

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
```