



U Y U N I

Client Configuration Guide

Uyuni '2020.07'

July 24, 2020

Table of Contents

Introduction	1
Supported Clients and Features	2
Supported Client Systems	2
Supported Tools Packages	3
Supported SUSE and openSUSE Client Features	4
Supported SUSE Linux Enterprise Server with Expanded Support Features	7
Supported Red Hat Enterprise Linux Features	9
Supported CentOS Features	12
Supported Oracle Features	15
Supported Ubuntu Features	17
Supported Debian Features	19
Software Channels	22
Packages Provided by SUSE Package Hub	22
Packages Provided by Appstream	22
Software Repositories	23
Local Repository Locations	23
Software Products	23
Using GPG Keys	25
Activation Keys	26
Combining Activation Keys	27
Activation Key Best Practices	28
Bootstrap Repository	31
Prepare to Create a Bootstrap Repository	31
Options for Automatic Mode	31
Configure Bootstrap Data File	32
Manually Generate a Bootstrap Repository	32
Repositories with Multiple Architectures	33
Bootstrap and Custom Channels	33
Contact Methods	34
SUSE Manager Daemon (rhnsd)	34
Configure rhnsd	35
Push via SSH	36
Push via Salt SSH	40
Available Parameters	41
Action Execution	41
Future Features	42
OSAD	42
Client Registration	45
Register Clients with the Web UI	45
Register Clients with a Bootstrap Script	46
Create a Bootstrap Script	46
Editing a Bootstrap Script	47
Connect Clients	48
Register on the Command Line (Salt)	49
SUSE Client Registration	51
Registering openSUSE Clients	51

Add Software Channels	51
Check Synchronization Status	51
Trust GPG Keys on Clients	52
Register Clients.	52
Registering SUSE Linux Enterprise Server with Expanded Support Clients	53
Add Software Channels	53
Check Synchronization Status	56
Register Expanded Support Clients.	57
Red Hat Client Registration	58
Registering Red Hat Enterprise Linux Clients with CDN	58
Import Entitlements and Certificates	58
Prepare Custom Repositories and Channels	60
Add Software Channels	62
Check Synchronization Status	63
Trust GPG Keys on Clients	65
Register Clients.	65
Package Management and Red Hat Enterprise Linux 8 Clients	66
Registering Red Hat Enterprise Linux Clients with RHUI.	66
Import Entitlements and Certificates	66
Prepare Custom Repositories and Channels	68
Add Software Channels	70
Check Synchronization Status	71
Trust GPG Keys on Clients	71
Register Clients.	72
Package Management and Red Hat Enterprise Linux 8 Clients	72
CentOS Client Registration	73
Registering CentOS Clients	73
Add Software Channels	73
Check Synchronization Status	74
Create an Activation Key	74
Trust GPG Keys on Clients	74
Register Clients.	75
Manage Errata	75
Oracle Client Registration	77
Registering Oracle Linux Clients	77
Add Software Channels	77
Check Synchronization Status	78
Create an Activation Key	78
Trust GPG Keys on Clients	78
Register Clients.	79
Ubuntu Client Registration	80
Registering Ubuntu 20.04 Clients	80
Add Software Channels	80
Check Synchronization Status	81
Trust GPG Keys on Clients	81
Root Access	82
Register Clients.	82
Registering Ubuntu 16.04 and 18.04 Clients.	82
Add Software Channels	83

Check Synchronization Status	83
Trust GPG Keys on Clients	84
Root Access	84
Register Clients.....	85
Registering Debian Clients	85
Prepare to Register	85
Add Software Channels	86
Check Synchronization Status	86
Trust GPG Keys on Clients	87
Register Clients.....	87
Client Registration on a Proxy	88
Register Clients to a Proxy in the Web UI	88
Register on the Command Line (Salt).....	89
Registering with a Bootstrap Script (Salt and Traditional).....	90
Automate Client Installation	91
Preparation	91
Upload a Profile	92
Variables and Snippets	93
SUSE Manager Proxy.....	94
Autoinstallation with AutoYaST	94
Bare Metal Provisioning	94
API Provisioning	95
Advanced PXE Installation Configuration	95
Kickstart	96
Before you Begin	96
Build a Bootable ISO	96
Integrating with PXE	97
Cobbler	98
Cobbler Requirements	98
Configure Cobbler	99
TFTP	100
Synchronize and Start the Cobbler Service.....	102
Autoinstallation Templates	102
Build ISOs with Cobbler.....	104
Bare Metal Provisioning	105
Patch Management	107
Create Patches.....	107
Apply Patches to Clients	108
System Locking	110
System Locks on Traditional Clients	110
System Locks on Salt Clients	110
Package Locks.....	111
Client Upgrades	112
Virtualization	113
Virtualization with Xen and KVM	113
Host Setup	113
Autoinstallation	114
Manage VM Guests	118
Virtualization with VMWare	119

VHM Setup	119
Troubleshooting SSL Errors on VMWare.....	120
Virtualization with Other Third Party Providers	120
Clusters	123
Autoinstallation Profile	123
Elect a Management Node.....	123
Manage Clusters	124
Manage Nodes.....	125
Upgrade Clusters	126
Virtual Host Managers	127
SUSE Support and VM Zones.....	127
VHM and Amazon Web Services	128
Create an Amazon EC2 VHM	128
VHM and Azure	128
Create an Azure VHM	129
Assigning Permissions	129
Azure UUID.....	130
VHM and Google Compute Engine.....	130
Create a GCE VHM	130
Assigning Permissions	131
GCE UUID	132
VHM and Kubernetes	132
Create a Kubernetes VHM	132
Retrieve Image Runtime Data.....	133
Permissions and Certificates	135
VHM and SUSE CaaS Platform	136
Onboarding CaaSP Nodes	136
Autoinstallation Profile of a SUSE CaaS Platform 4 Node.....	137
Manage a SUSE CaaS Platform Cluster With Uyuni	137
Troubleshooting Clients	141
Bare Metal Systems.....	141
Cloned Salt Clients	141
Disabling the FQDNS grain.....	142
Mounting /tmp with noexec.....	142
Passing Grains to a Start Event	142
Proxy Connections and FQDN	143
Registering Older Clients.....	143
AutoYast Example File	145
Minimalist AutoYaST Profile for Automated Installations and Useful Enhancements.....	145
GNU Free Documentation License	147

Introduction

Publication Date: 2020-07-24

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

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

For a complete list of supported clients and features, see [[Client-configuration > Supported-features >](#)].

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

Supported Clients and Features

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

This section contains summary of supported client systems. For a detailed list of features available on each client, see the following pages.

Supported Client Systems

Supported operating systems for traditional and Salt clients are listed in this table.

The icons in this table indicate:

- ✓ clients running this operating system are supported by SUSE
- ✗ clients running this operating system are not supported by SUSE
- ? clients are under consideration, and may or may not be supported at a later date.



Client operating system versions and SP levels must be under general support (normal or LTSS) to be supported with Uyuni. For details on supported product versions, see <https://www.suse.com/lifecycle>.



The operating system you run on a client is supported by the organization that supplies the operating system.

Table 1. Supported Client Systems

Operating System	Architecture	Traditional Clients	Salt Clients
SUSE Linux Enterprise 15	x86_64, POWER, IBM Z, ARM	✓	✓
SUSE Linux Enterprise 12	x86_64, POWER, IBM Z, ARM	✓	✓
SUSE Linux Enterprise 11	x86, x86_64, Itanium, IBM POWER, IBM Z	✓	✓
SUSE Linux Enterprise Server for SAP	x86_64, POWER	✓	✓
openSUSE Leap 15	x86_64	✓	✓
SUSE Linux Enterprise Server ES 8	x86_64	✗	✓

Operating System	Architecture	Traditional Clients	Salt Clients
SUSE Linux Enterprise Server ES 7	x86_64	✓	✓
SUSE Linux Enterprise Server ES 6	x86, x86_64	✓	✓
Red Hat Enterprise Linux 8	x86_64	✗	✓
Red Hat Enterprise Linux 7	x86_64	✓	✓
Red Hat Enterprise Linux 6	x86, x86_64	✓	✓
Oracle Linux 8	x86_64	✗	✓
Oracle Linux 7	x86_64	✓	✓
Oracle Linux 6	x86, x86_64	✓	✓
CentOS 8	x86_64	✗	✓
CentOS 7	x86_64	✓	✓
CentOS 6	x86, x86_64	✓	✓
Ubuntu 18.04	x86_64	✗	✓
Ubuntu 16.04	x86_64	✗	✓
Debian 9	x86_64	✗	✓
Debian 10	x86_64	✗	✓

Supported Tools Packages

The **spacewalk-utils** and **spacewalk-utils-extras** packages can provide additional services and features. The **spacewalk-utils** package in Uyuni 4.1 and later is fully supported by SUSE, and contains these tools:

Table 2. Spacewalk Utilities

Tool Name	Description	Supported?
spacewalk-common-channels	Add channels not provided by SUSE Customer Center	✓
spacewalk-hostname-rename	Change the hostname of the Uyuni Server	✓

Tool Name	Description	Supported?
<code>spacewalk-clone-by-date</code>	Clone channels by a specific date	✓
<code>spacewalk-sync-setup</code>	Set up ISS master and slave organization mappings	✓
<code>spacewalk-manage-channel-lifecycle</code>	Manage channel lifecycles	✓

The `spacewalk-utils-extras` package in Uyuni 4.1 and later has limited support by SUSE.

Supported SUSE and openSUSE Client Features

This table lists the availability of various features on SUSE and openSUSE clients. This table covers all variants of the SUSE Linux Enterprise operating system, including SLES, SLED, SUSE Linux Enterprise Server for SAP, and SUSE Linux Enterprise Server for HPC.



The operating system you run on a client is supported by the organization that supplies the operating system. SUSE Linux Enterprise is supported by SUSE. openSUSE is supported by the SUSE community.

The icons in this table indicate:

- ✓ the feature is available on both Salt and traditional clients
- ✗ the feature is not available
- ? the feature is under consideration, and may or may not be made available at a later date
- Traditional the feature is supported only on traditional clients
- Salt the feature is supported only on Salt clients.

Table 3. Supported Features on SUSE and openSUSE Operating Systems

Feature	SUSE Linux Enterprise 11	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15.1
Client	✓	✓	✓	✓
System packages	SUSE	SUSE	SUSE	openSUSE Community
Registration	✓	✓	✓	Salt
Install packages	✓	✓	✓	Salt
Apply patches	✓	✓	✓	Salt
Remote commands	✓	✓	✓	Salt

Feature	SUSE Linux Enterprise 11	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15.1
System package states	Salt	Salt	Salt	Salt
System custom states	Salt	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt	Salt
System set manager (SSM)	✓	✓	✓	Salt
Service pack migration	✓	✓	✓	Salt
Basic Virtual Guest Management *	Traditional	✓	✓	Salt
Advanced Virtual Guest Management *	✗	Salt	Salt	Salt
Virtual Guest Installation (AutoYaST), as Host OS	Traditional	Traditional	Traditional	✗
Virtual Guest Installation (image template), as Host OS	✗	Salt	Salt	Salt
Virtual Guest Management	✗	Salt	Salt	Salt
System deployment (PXE/AutoYaST)	✓	✓	✓	✓
System redeployment (AutoYaST)	Traditional	✓	✓	Salt

Feature	SUSE Linux Enterprise 11	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15.1
Contact methods	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✓	✓	✓	Salt
Action chains	✓	✓	✓	Salt
Software crash reporting	✗	✗	✗	✗
Staging (pre-download of packages)	✓	✓	✓	Salt
Duplicate package reporting	✓	✓	✓	Salt
CVE auditing	✓	✓	✓	Salt
SCAP auditing	✓	✓	✓	Salt
Package verification	Traditional	Traditional	Traditional	✗
Package locking	Traditional	Traditional	Traditional	✗
System locking	Traditional	Traditional	Traditional	✗
System snapshot	Traditional	Traditional	Traditional	✗
Configuration file management	✓	✓	✓	Salt
Package profiles	Traditional. Salt: Profiles supported, Sync not supported	Traditional. Salt: Profiles supported, Sync not supported	Traditional. Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓	✓	✓
Monitoring	?	Salt	Salt	Salt
Docker buildhost	✗	Salt	Salt	?
Build Docker image with OS	✗	Salt	Salt	Salt
Kiwi buildhost	✗	Salt	?	?

Feature	SUSE Linux Enterprise 11	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15.1
Build Kiwi image with OS	✗	Salt	?	✗

* Virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported SUSE Linux Enterprise Server with Expanded Support Features

This table lists the availability of various features on SUSE Linux Enterprise Server with Expanded Support clients.



The operating system you run on a client is supported by the organization that supplies the operating system. SUSE Linux Enterprise Server with Expanded Support is supported by SUSE.

The icons in this table indicate:

- ✓ the feature is available on both Salt and traditional clients
- ✗ the feature is not available
- ? the feature is under consideration, and may or may not be made available at a later date
- Traditional the feature is supported only on traditional clients
- Salt the feature is supported only on Salt clients.

Table 4. Supported Features on SUSE Linux Enterprise Server with Expanded Support Operating Systems

Feature	SLES ES 6	SLES ES 7	SLES ES 8
Client	✓	✓	Salt
System packages	SUSE	SUSE	SUSE
Registration	✓	✓	Salt
Install packages	✓	✓	Salt

Feature	SLES ES 6	SLES ES 7	SLES ES 8
Apply patches	✓	✓	Salt
Remote commands	✓	✓	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt
System set manager (SSM)	Salt	Salt	Salt
Service pack migration	N/A	N/A	N/A
Basic Virtual Guest Management *	Traditional	✓	Salt
Advanced Virtual Guest Management *	✗	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	Traditional	Traditional	✗
Virtual Guest Installation (image template), as Host OS	Traditional	✓	Salt
System deployment (PXE/Kickstart)	✓	✓	Salt
System redeployment (Kickstart)	Traditional	✓	✗
Contact methods	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✓	✓	Salt
Action chains	✓	✓	Salt
Software crash reporting	✗	Traditional	✗
Staging (pre-download of packages)	✓	✓	Salt

Feature	SLES ES 6	SLES ES 7	SLES ES 8
Duplicate package reporting	✓	✓	Salt
CVE auditing	✓	✓	Salt
SCAP auditing	✓	✓	Salt
Package verification	Traditional	Traditional	✗
Package locking	Traditional	Traditional	✗
System locking	Traditional	Traditional	✗
System snapshot	Traditional	Traditional	Salt
Configuration file management	✓	✓	Salt
Snapshots and profiles	Traditional. Salt: Profiles supported, Sync not supported	Traditional. Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓	Salt
Monitoring	Salt	Salt	Salt
Docker buildhost	✗	✗	✗
Build Docker image with OS	?	?	?
Kiwi buildhost	✗	✗	✗
Build Kiwi image with OS	✗	✗	✗

* Virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported Red Hat Enterprise Linux Features

This table lists the availability of various features on native Red Hat Enterprise Linux clients (without Expanded Support).



The operating system you run on a client is supported by the organization that supplies the operating system. Red Hat Enterprise Linux is supported by Red Hat.

The icons in this table indicate:

- ✓ the feature is available on both Salt and traditional clients
- ✗ the feature is not available
- ? the feature is under consideration, and may or may not be made available at a later date
- Traditional the feature is supported only on traditional clients
- Salt the feature is supported only on Salt clients.

Table 5. Supported Features on Red Hat Enterprise Linux Operating Systems

Feature	RHEL 6	RHEL 7	RHEL 8
Client	✓	✓	Salt
System packages	Red Hat	Red Hat	Red Hat
Registration	✓	✓	Salt
Install packages	✓	✓	Salt
Apply patches	✓	✓	Salt
Remote commands	✓	✓	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt
System set manager (SSM)	Salt	Salt	Salt
Service pack migration	N/A	N/A	N/A
Basic Virtual Guest Management *	Traditional	✓	Salt
Advanced Virtual Guest Management *	✗	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	Traditional	Traditional	✗

Feature	RHEL 6	RHEL 7	RHEL 8
Virtual Guest Installation (image template), as Host OS	Traditional	✓	Salt
System deployment (PXE/Kickstart)	✓	✓	Salt
System redeployment (Kickstart)	Traditional	✓	Salt
Contact methods	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✓	✓	Salt
Action chains	✓	✓	Salt
Software crash reporting	✗	Traditional	✗
Staging (pre-download of packages)	✓	✓	Salt
Duplicate package reporting	✓	✓	Salt
CVE auditing	✓	✓	Salt
SCAP auditing	✓	✓	Salt
Package verification	Traditional	Traditional	✗
Package locking	Traditional	Traditional	✗
System locking	Traditional	Traditional	✗
System snapshot	Traditional	Traditional	✗
Configuration file management	✓	✓	Salt
Snapshots and profiles	Traditional. Salt: Profiles supported, Sync not supported	Traditional. Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓	Salt
Monitoring	Salt	Salt	Salt
Docker buildhost	✗	✗	✗

Feature	RHEL 6	RHEL 7	RHEL 8
Build Docker image with OS	?	?	?
Kiwi buildhost	✗	✗	✗
Build Kiwi image with OS	✗	✗	✗

* Virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported CentOS Features

This table lists the availability of various features on CentOS clients.



The operating system you run on a client is supported by the organization that supplies the operating system. CentOS is supported by the CentOS community.

The icons in this table indicate:

- ✓ the feature is available on both Salt and traditional clients
- ✗ the feature is not available
- ? the feature is under consideration, and may or may not be made available at a later date
- Traditional the feature is supported only on traditional clients
- Salt the feature is supported only on Salt clients.

Table 6. Supported Features on CentOS Operating Systems

Feature	CentOS 6	CentOS 7	CentOS 8
Client	✓ (plain CentOS)	✓ (plain CentOS)	Salt (plain CentOS)
System packages	CentOS Community	CentOS Community	CentOS Community
Registration	✓	✓	Salt
Install packages	✓	✓	Salt

Feature	CentOS 6	CentOS 7	CentOS 8
Apply patches (requires CVE ID)	✓ (third-party service required for errata)	✓ (third-party service required for errata)	Salt (third-party service required for errata)
Remote commands	✓	✓	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt
System set manager (SSM)	✓	✓	Salt
Service pack migration	N/A	N/A	N/A
Basic Virtual Guest Management *	Traditional	✓	Salt
Advanced Virtual Guest Management *	✗	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	Traditional	Traditional	✗
Virtual Guest Installation (image template), as Host OS	Traditional	✓	Salt
System deployment (PXE/Kickstart)	✓	✓	Salt
System redeployment (Kickstart)	Traditional	✓	Salt
Contact methods	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✓	✓	Salt
Action chains	✓	✓	Salt
Software crash reporting	✗	Traditional	✗
Staging (pre-download of packages)	✓	✓	Salt

Feature	CentOS 6	CentOS 7	CentOS 8
Duplicate package reporting	✓	✓	Salt
CVE auditing (requires CVE ID)	✓	✓	Salt
SCAP auditing	✓	✓	Salt
Package verification	Traditional	Traditional	✗
Package locking	Traditional	Traditional	✗
System locking	Traditional	Traditional	✗
System snapshot	Traditional	Traditional	✗
Configuration file management	✓	✓	Salt
Snapshots and profiles	Traditional. Salt: Profiles supported, Sync not supported	Traditional. Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓	Salt
Monitoring	Salt	Salt	Salt
Docker buildhost	✗	✗	✗
Build Docker image with OS	✗	✗	✗
Kiwi buildhost	✗	✗	✗
Build Kiwi image with OS	✗	✗	✗

* Virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported Oracle Features

This table lists the availability of various features on Oracle Linux clients.



The operating system you run on a client is supported by the organization that supplies the operating system. Oracle Linux is supported by Oracle.

The icons in this table indicate:

- ✓ the feature is available on both Salt and traditional clients
- ✕ the feature is not available
- ? the feature is under consideration, and may or may not be made available at a later date
- Traditional the feature is supported only on traditional clients
- Salt the feature is supported only on Salt clients

Table 7. Supported Features on Oracle Linux Operating Systems

Feature	Oracle Linux 6	Oracle Linux 7	Oracle Linux 8
Client	✓	✓	Salt
Operating system packages	✓	✓	Salt
Registration	✓	✓	Salt
Install packages	✓	✓	Salt
Apply patches (requires CVE ID)	✓	✓	Salt
Remote commands	✓	✓	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt
System set manager (SSM)	✓	✓	Salt
Service pack migration	N/A	N/A	N/A
Basic Virtual Guest Management ✱	Traditional	✓	Salt

Feature	Oracle Linux 6	Oracle Linux 7	Oracle Linux 8
Advanced Virtual Guest Management *	✗	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	Traditional	Traditional	✗
Virtual Guest Installation (image template), as Host OS	Traditional	✓	Salt
System deployment (PXE/Kickstart)	✓	✓	Salt
System redeployment (Kickstart)	Traditional	✓	Salt
Contact methods	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✓	✓	Salt
Action chains	✓	✓	Salt
Software crash reporting	✓	✓	Salt
Staging (pre-download of packages)	✓	✓	Salt
Duplicate package reporting	✓	✓	Salt
CVE auditing (requires CVE ID)	✓	✓	Salt
SCAP auditing	✓	✓	Salt
Package verification	Traditional	Traditional	✗
Package locking	Traditional	Traditional	✗
System locking	Traditional	Traditional	✗
System snapshot	Traditional	Traditional	✗
Configuration file management	✓	✓	Salt
Snapshots and profiles	Traditional. Salt: Profiles supported, Sync not supported	Traditional. Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported

Feature	Oracle Linux 6	Oracle Linux 7	Oracle Linux 8
Power management	✓	✓	Salt
Monitoring	Salt	Salt	Salt
Docker buildhost	✗	✗	✗
Build Docker image with OS	✗	✗	✗
Kiwi buildhost	✗	✗	✗
Build Kiwi image with OS	✗	✗	✗

* Virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported Ubuntu Features

This table lists the availability of various features on Ubuntu clients.



The operating system you run on a client is supported by the organization that supplies the operating system. Ubuntu is supported by Canonical.

The icons in this table indicate:

- ✓ the feature is available on both Salt and traditional clients
- ✗ the feature is not available
- ? the feature is under consideration, and may or may not be made available at a later date
- Traditional the feature is supported only on traditional clients
- Salt the feature is supported only on Salt clients.

Table 8. Supported Features on Ubuntu Operating Systems

Feature	Ubuntu 16.04	Ubuntu 18.04
Client	✓	✓

Feature	Ubuntu 16.04	Ubuntu 18.04
System packages	Canonical	Canonical
Registration	Salt	Salt
Install packages	Salt	Salt
Apply patches	?	?
Remote commands	Salt	Salt
System package states	Salt	Salt
System custom states	Salt	Salt
Group custom states	Salt	Salt
Organization custom states	Salt	Salt
System set manager (SSM)	Salt	Salt
Service pack migration	N/A	N/A
Basic Virtual Guest Management *	Salt	Salt
Advanced Virtual Guest Management *	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗	✗
Virtual Guest Installation (image template), as Host OS	Salt	Salt
System deployment (PXE/Kickstart)	✗	✗
System redeployment (Kickstart)	✗	✗
Contact methods	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt	Salt
Action chains	Salt	Salt
Software crash reporting	✗	✗
Staging (pre-download of packages)	Salt	Salt
Duplicate package reporting	Salt	Salt
CVE auditing	?	?

Feature	Ubuntu 16.04	Ubuntu 18.04
SCAP auditing	?	?
Package verification	✗	✗
Package locking	✗	✗
System locking	✗	✗
System snapshot	✗	✗
Configuration file management	Salt	Salt
Package profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓
Monitoring	✗	Salt
Docker buildhost	?	?
Build Docker image with OS	Salt	Salt
Kiwi buildhost	✗	✗
Build Kiwi image with OS	✗	✗

* Virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported Debian Features

This table lists the availability of various features on Debian clients.



The operating system you run on a client is supported by the organization that supplies the operating system. Debian is supported by the Debian community.

The icons in this table indicate:

- ✓ the feature is available on both Salt and traditional clients
- ✗ the feature is not available

- **?** the feature is under consideration, and may or may not be made available at a later date
- Traditional the feature is supported only on traditional clients
- Salt the feature is supported only on Salt clients.

Table 9. Supported Features on Debian Operating Systems

Feature	Debian 9	Debian 10
Client	✓	✓
System packages	Debian Community	Debian Community
Registration	Salt	Salt
Install packages	Salt	Salt
Apply patches	?	?
Remote commands	Salt	Salt
System package states	Salt	Salt
System custom states	Salt	Salt
Group custom states	Salt	Salt
Organization custom states	Salt	Salt
System set manager (SSM)	Salt	Salt
Service pack migration	N/A	N/A
Basic Virtual Guest Management *	Salt	Salt
Advanced Virtual Guest Management *	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗	✗
Virtual Guest Installation (image template), as Host OS	Salt	Salt
System deployment (PXE/Kickstart)	✗	✗
System redeployment (Kickstart)	✗	✗
Contact methods	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt	Salt
Action chains	Salt	Salt

Feature	Debian 9	Debian 10
Software crash reporting	✗	✗
Staging (pre-download of packages)	Salt	Salt
Duplicate package reporting	Salt	Salt
CVE auditing	?	?
SCAP auditing	?	?
Package verification	✗	✗
Package locking	✗	✗
System locking	✗	✗
System snapshot	✗	✗
Configuration file management	Salt	Salt
Package profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓
Monitoring	Salt	Salt
Docker buildhost	?	?
Build Docker image with OS	Salt	Salt
Kiwi buildhost	✗	✗
Build Kiwi image with OS	✗	✗

* Virtual Guest Management:

In this table, virtual guest management is split into basic and advanced.

Basic virtual guest management includes listing VMs, slow refresh, VM lifecycle actions (start, stop, resume, pause), and modifying VM vCPU and Memory.

Advanced virtual guest management includes fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Software Channels

Channels are a method of grouping software packages. Software packages are provided by repositories, and repositories are associated with channels. Subscribing a client to a software channel allows the client to install and update any of the software associated with it.

In Uyuni, channels are divided into base channels and child channels. Organizing channels in this way ensures that only compatible packages are installed on each system. A client must be subscribed to only one base channel, assigned during registration based on the client operating system and architecture. For paid channels provided by a vendor, you must have an associated subscription.

A base channel consists of packages built for a specific operating system type, version, and architecture. For example, the SUSE Linux Enterprise Server 15 x86_64 base channel contains only software compatible with that operating system and architecture.

A child channel is associated with a base channel and provides only packages that are compatible with the base channel. A system can be subscribed to multiple child channels of its base channel. When a system has been assigned to a base channel, it is only possible for that system to install the related child channels. For example, if a system has been assigned to the SUSE Linux Enterprise Server 15 x86_64 base channel, they will only be able to install or update packages made available through the compatible base channel, or any of its associated child channels.

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

For more on using channels, including custom channels, see [**Administration > Channel-management >**].

Packages Provided by SUSE Package Hub

SUSE Package Hub is an extension to SUSE Linux Enterprise products that provides additional open source software provided by the openSUSE community. If you are using SUSE Linux Enterprise operating systems on your clients, you can enable the SUSE Package Hub extension to access these additional packages. This provides the SUSE Package Hub channels, which you can subscribe your clients to.

On Salt clients, you must apply the highstate after subscribing to the channel to be able to install packages.

Packages Provided by Appstream

Appstream is a modular package repository that provides additional software for most non-SUSE Linux distributions, including Red Hat and Oracle Linux. In most cases, you will need to install the base channel for your distribution, and the appropriate Appstream channel to ensure that you have all the software you need.

If you need to transform Appstream repositories into non-modular repositories for use with some upgrade operations, you can use the Appstream filter with the content lifecycle management tool. For more

information about the Appstream filter, see [**Administration > Content-lifecycle-examples >**].

Software Repositories

Repositories are used to collect software packages. When you have access to a software repository, you can install any of the software that the repository provides. You must have at least one repository associated with your software channels in Uyuni in order to assign clients to the channel and install and update packages on the client.

Most default channels in Uyuni are already associated with the correct repositories. If you are creating custom channels, you will need to associate a repository that you have access to, or that you have created yourself.

For more information about custom repositories and channels, see [**Administration > Custom-channels >**].

Local Repository Locations

You can configure local repositories on Salt clients, in order to provide packages that are not supplied by Uyuni channels.

Local repositories are disabled during onboarding. After a client has completed onboarding, you can add local repositories to these locations:

Table 10. Local Repository Locations

Client Operating System	Local Repository Directory
SUSE Linux Enterprise Server	<code>/etc/zypp/repos.d</code>
openSUSE	<code>/etc/zypp/repos.d</code>
SUSE Linux Enterprise Server Expanded Support	<code>/etc/yum.repos.d/</code>
Red Hat Enterprise Linux	<code>/etc/yum.repos.d/</code>
CentOS	<code>/etc/yum.repos.d/</code>
Ubuntu	<code>/etc/apt/sources.list.d/</code>
Debian	<code>/etc/apt/sources.list.d/</code>

For Salt clients, local repositories remain persistent, even when applying the highstate.

Software Products

In Uyuni, software is made available in products. Your SUSE subscription allows you to access a range of different products, which you can browse and select in the Uyuni Web UI by navigating to **Admin > Setup Wizard > Products**.

Products contain any number of software channels. Click the **Show product's channels** icon to see the channels included in the product. When you have added a product and synchronized successfully, you have access to the channels provided by the product, and can use the packages in the product on your Uyuni Server and clients.

Procedure: Adding Software Channels

1. In the Uyuni Web UI, navigate to **Admin > Setup Wizard > Products**.
2. Locate the appropriate products for your client operating system and architecture using the search bar, and check the appropriate product. This will automatically check all required channels. Click the arrow to see the complete list of related products, and ensure that any extra products you require are checked.
3. Click [**Add Products**] and wait until the products have finished synchronizing.

Using GPG Keys

Uyuni uses GPG keys to check the authenticity of software packages installed on clients. In many cases, you will not need to adjust the GPG settings to be able to install software on your clients.

By default, some operating systems do not trust the GPG key for the Uyuni client tools. The clients can be successfully bootstrapped without the GPG key being trusted. However, you will not be able to install new client tool packages or update them until the keys are trusted.

Procedure: Trusting GPG Keys on Clients

1. On the Uyuni Server, at the command prompt, check the contents of the `/srv/www/htdocs/pub/` directory. This directory contains all available public keys. Take a note of the key that applies to the client you are registering.
2. Open the relevant bootstrap script, locate the `ORG_GPG_KEY=` parameter and add the required key. For example:

```
uyuni-gpg-pubkey-0d20833e.key
```

You do not need to delete any previously stored keys.

3. If you are bootstrapping clients from the Uyuni Web UI, you will need to use a Salt state to trust the key. Create the Salt state and assign it to the organization. You can then use an activation key and configuration channels to deploy the key to the clients.

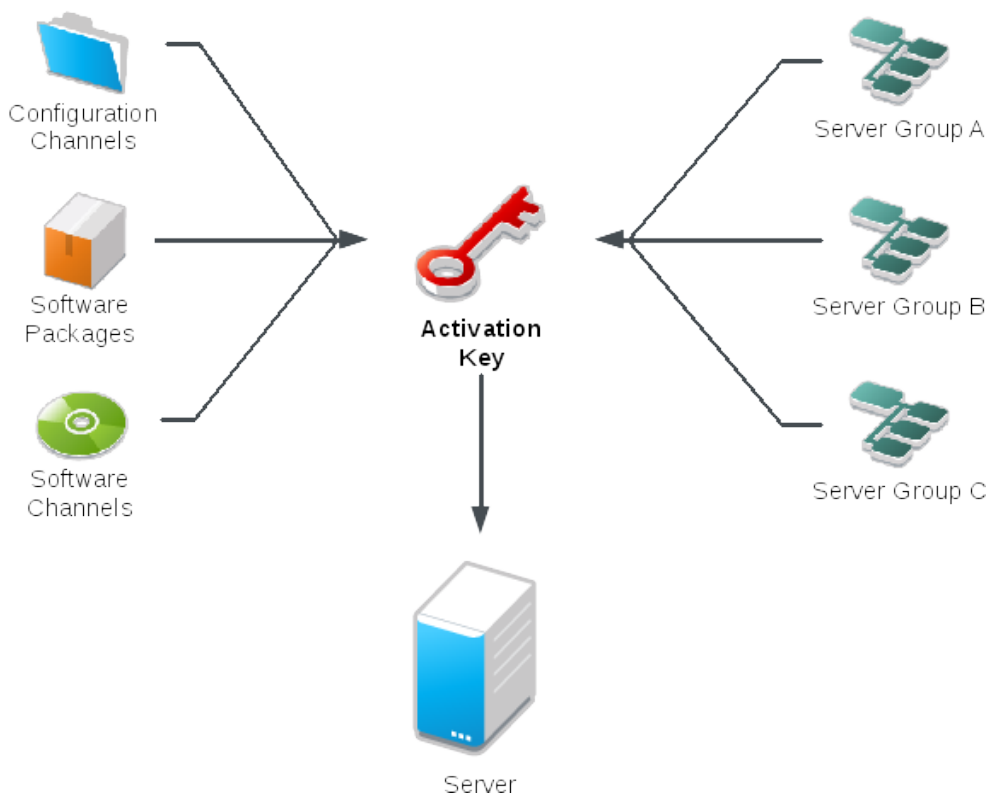
Activation Keys

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

In Uyuni, an activation key is a group of configuration settings with a label. You can apply all configuration settings associated with an activation key by adding its label as a parameter to a bootstrap script. We recommend you use an activation key label in combination with a bootstrap script. When the bootstrap script is executed all configuration settings associated with the label are applied to the system the script is run on.

An activation key can specify:

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



Procedure: Creating an Activation Key

1. In the Uyuni Web UI, as an administrator, navigate to **Systems > Activation Keys**.

2. Click the **[Create Key]** button.
3. On the **Activation Key Details** page, in the **Description** field, enter a name for the activation key.
4. In the **Key** field, enter the distribution and service pack associated with the key. For example, **SLES12-SP4** for SUSE Linux Enterprise Server 12 SP4.



Do not use commas in the **Key** field for any SUSE products. However, you **must** use commas for Red Hat Products. For more information, see [**Reference > Systems >**].

5. In the **Base Channels** drop-down box, select the appropriate base software channel, and allow the relevant child channels to populate. For more information, see [reference:admin/setup-wizard.pdf](#) and [**Administration > Custom-channels >**].
6. Select the child channels you need (for example, the mandatory SUSE Manager tools and updates channels).
7. We recommend you leave the **Contact Method** set to **Default**.
8. We recommend you leave the **Universal Default** setting unchecked.
9. Click **[Create Activation Key]** to create the activation key.
10. Check the **Configuration File Deployment** check box to enable configuration management for this key, and click **[Update Activation Key]** to save this change.

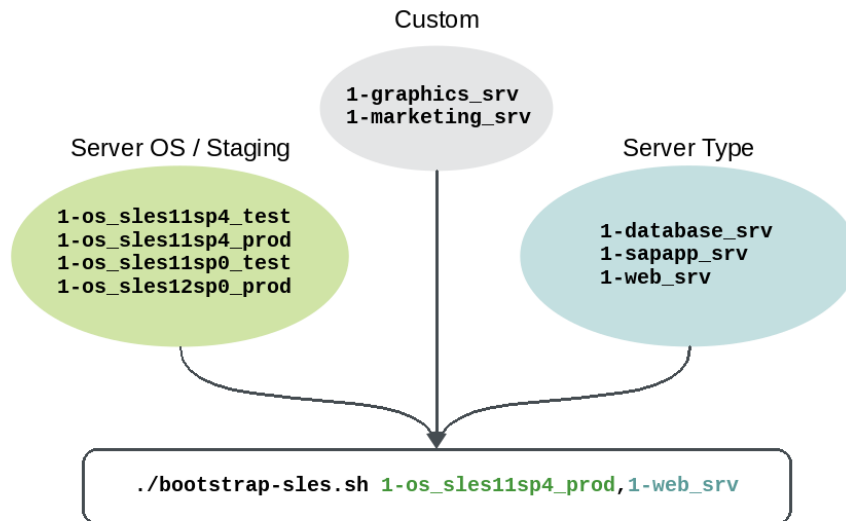


The **Configuration File Deployment** check box does not appear until after you have created the activation key. Ensure you go back and check the box if you need to enable configuration management.

Combining Activation Keys

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

Combining Activation Keys



Combining Activation Keys

Server OS / Stage Key

Base Channels:

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

The following child channels of `sles12sp0_3prod-sles12-pool-x86_64 (01.06.2015)` can be associated with this activation key.

- sles12sp0_3prod-obs-home-packages-x86_64
- sles12sp0_3prod-obs-server-packages-x86_64
- sles12sp0_3prod-sle-12-ga-desktop-amd-driver-x86_64-we
- sles12sp0_3prod-sle-12-ga-desktop-nvidia-driver-x86_64-we
- sles12sp0_3prod-sle-ha12-pool-x86_64
- sles12sp0_3prod-sle-ha12-updates-x86_64
- sles12sp0_3prod-sle-manager-tools12-pool-x86_64
- sles12sp0_3prod-sle-manager-tools12-updates-x86_64
- sles12sp0_3prod-sle-module-adv-systems-management12-pool-x86_64
- sles12sp0_3prod-sle-module-adv-systems-management12-updates-x86_64
- sles12sp0_3prod-sle-module-legacy12-pool-x86_64
- sles12sp0_3prod-sle-module-legacy12-updates-x86_64
- sles12sp0_3prod-sle-module-public-cloud12-pool-x86_64
- sles12sp0_3prod-sle-module-public-cloud12-updates-x86_64
- sles12sp0_3prod-sle-module-web-scripting12-pool-x86_64
- sles12sp0_3prod-sle-module-web-scripting12-updates-x86_64
- sles12sp0_3prod-sles12-updates-x86_64
- sles12sp0_3prod-sle-sdk12-pool-x86_64
- sles12sp0_3prod-sle-sdk12-updates-x86_64
- sles12sp0_3prod-sle-we12-pool-x86_64

Update Key

Any other type of key

Base Channels:

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

- sles12sp0_3prod-sle-module-legacy12-pool-x86_64
- sles12sp0_3prod-sle-module-legacy12-updates-x86_64
- sles12sp0_3prod-sle-module-public-cloud12-pool-x86_64
- sles12sp0_3prod-sle-module-public-cloud12-updates-x86_64
- sles12sp0_3prod-sle-module-web-scripting12-pool-x86_64
- sles12sp0_3prod-sle-module-web-scripting12-updates-x86_64
- sles12sp0_3prod-sles12-updates-x86_64
- sles12sp0_3prod-sle-sdk12-pool-x86_64
- sles12sp0_3prod-sle-sdk12-updates-x86_64
- sles12sp0_3prod-sle-we12-pool-x86_64
- sles12sp0_3prod-sle-we12-updates-x86_64
- SUSE-Manager-Proxy-1.7-Pool for x86_64
- SLES11-SP1-Pool for x86_64 Proxy 1.7
- SLES11-SP1-Updates for x86_64 Proxy 1.7
- SLES11-SP2-Core for x86_64 Proxy 1.7
- SLES11-SP2-Updates for x86_64 Proxy 1.7
- SUSE-Manager-Proxy-1.7-Updates for x86_64
- SUSE-Manager-Proxy-2.1-Pool for x86_64
- SLES11-SP3-Pool for x86_64 Proxy 2.1
- SLES11-SP3-Updates for x86_64 Proxy 2.1
- SUSE-Manager-Proxy-2.1-Updates for x86_64

Update Key



With Salt clients, you cannot combine activation keys. Only the first key will be used.

Activation Key Best Practices

Default Parent Channel

Avoid using the **SUSE Manager Default** parent channel. This setting forces Uyuni to choose a parent channel that best corresponds to the installed operating system, which can sometimes lead to unexpected

behavior. Instead, we recommend you create activation keys specific to each distribution and architecture.

Bootstrapping with Activation Keys

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

Bandwidth Requirements

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

These options create bandwidth usage:

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

Key Label Naming

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

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

When creating key labels, consider these tips:

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

Example activation key label names:

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

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



Do not use commas in the **Key** field for any SUSE products. However, you **must** use commas for Red Hat Products. For more information, see [**Reference > Systems >**].

Included Channels

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



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

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

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

Typical packages to be included:

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

Bootstrap Repository

A bootstrap repository contains packages for installing Salt on clients, as well as the required packages for registering Salt or traditional clients during bootstrapping. Bootstrap repositories are automatically created and regenerated on the Uyuni Server for every synchronized product.

Prepare to Create a Bootstrap Repository

When you select a product for synchronization, the bootstrap repository is automatically created as soon as all mandatory channels are fully mirrored.

Procedure: Checking Synchronization Progress

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

Procedure: Checking Synchronization Progress from the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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

Options for Automatic Mode

You can change how the automated bootstrap repository creation works. This section details the various settings.

Flush Mode

By default, every regeneration starts with an empty repository and copies only the latest packages into it. To disable this behavior, add or edit this value in **/etc/rhn/rhn.conf**:

```
server.susemanager.bootstrap_repo_flush = 0
```

Alternatively, you can use the **--no-flush** option when you create the bootstrap repository from the command prompt.

Automatic Mode

By default, automated regeneration of the bootstrap repositories is enabled. To disable it, add or edit this value in **/etc/rhn/rhn.conf**:

```
server.susemanager.auto_generate_bootstrap_repo = 0
```

Configure Bootstrap Data File

The tool uses a data file with information about which packages are required for each distribution. The data file is stored at `/usr/share/susemanager/mgr_bootstrap_data.py`. SUSE updates this file regularly. If you want to make changes to this file, do not edit it directly. Instead, create a copy in the same directory and edit your copy:

```
cd /usr/share/susemanager/
cp mgr_bootstrap_data.py my_data.py
```

When you have made your changes, configure Uyuni to use the new file. Add or edit this value in `/etc/rhn/rhn.conf`:

```
server.susemanager.bootstrap_repo_datamodule = my_data
```



On the next update, the new data from SUSE will overwrite the original data file, not the new one. You will need to keep the new file up to date with changes provided by SUSE.

Manually Generate a Bootstrap Repository

By default, bootstrap repositories are regenerated daily. You can manually create the bootstrap repository from the command prompt:

Procedure: Generating the Bootstrap Repository for SUSE Linux Enterprise

1. At the command prompt on the Uyuni Server, as root, list the available bootstrap repositories:

```
mgr-create-bootstrap-repo -l
```

2. Create the bootstrap repository, using the appropriate repository name as the product label:

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

The client repository is located in `/srv/www/htdocs/pub/repositories/`.

Procedure: Specifying a Parent Channel for a Bootstrap Repository

If you have mirrored more than one product (for example, SLES and SLES for SAP), or if you use custom channels, you need to specify the parent channel to use when creating the bootstrap repository.

1. Check which parent channels you have available:

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

2. Specify the appropriate parent channel:

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

Repositories with Multiple Architectures

If you are creating bootstrap repositories that include multiple different architectures, you need to be careful that all architectures are updated correctly. For example, the x86_64 and IBM Z architectures for SLE use the same bootstrap repository URL at </srv/www/htdocs/pub/repositories/sle/15/1/bootstrap/>.

Because the **flush** option is enabled by default, when you attempt to generate the bootstrap repository for multiple architectures, only one architecture would be generated. To avoid this, use the **--no-flush** option at the command prompt when creating additional architectures. For example:

```
mgr-create-bootstrap-repo -c SLE-15-SP1-x86_64
mgr-create-bootstrap-repo --no-flush -c SLE-15-SP1-s390x
```

Bootstrap and Custom Channels

If you are using custom channels, you can use the **--with-custom-channels** option with the **mgr-create-bootstrap-repo** command. In this case, you will also need to specify the parent channel to use.

Automatic creation of a bootstrap repository might fail if you are using custom channels. In this case, you will need to create the repository manually.

For more information about custom channels, see [**Administration > Custom-channels >**].

Contact Methods

There are a number of ways that the Uyuni Server can communicate with clients. Which one you use depends on your network architecture.

The Uyuni daemon (**rhnsd**) runs on traditional client systems and periodically connects with Uyuni to check for new updates and notifications. It does not apply to Salt clients.

Push via SSH and Push via Salt SSH are used in environments where clients cannot reach the Uyuni Server directly. In this environment, clients are located in a firewall-protected zone called a DMZ. No system within the DMZ is authorized to open a connection to the internal network, including the Uyuni Server.

OSAD is an alternative contact method between Uyuni and its clients. OSAD allows registered client systems to execute scheduled actions immediately.

SUSE Manager Daemon (rhnsd)

The Uyuni daemon (**rhnsd**) runs on traditional client systems and periodically connects with Uyuni to check for new updates and notifications. It does not apply to Salt clients.

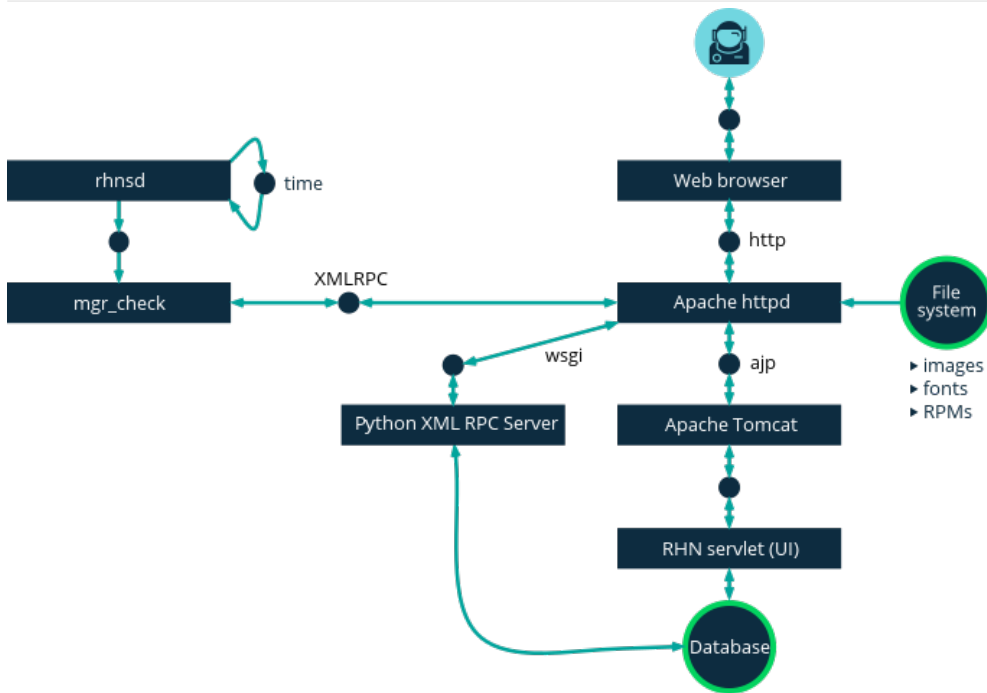
It is only used on SUSE Linux Enterprise 11 and Red Hat Enterprise Linux Server 6, as these systems do not use systemd. On later operating systems, a systemd timer (**rhnsd.timer**) is used and controlled by **rhnsd.service**.

Start the daemon with **/etc/init.d/rhnsd**.

By default, it will check every four hours for new actions. This means it can take some time for clients to execute scheduled actions.

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

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



Configure rhnsd

The **rhnsd** initialization script has a configuration file on the client system at **/etc/sysconfig/rhn/rhnsd**.

An important parameter for the daemon is its check-in frequency. The default interval time is four hours (240 minutes). The minimum allowed time interval is one hour (60 minutes). If you set the interval below one hour, it will change back to the default of 4 hours (240 minutes).

If you modify the **rhnsd** configuration file, execute this command as root to restart the daemon and pick up your changes:

```
/etc/init.d/rhnsd restart
```

To see the status of **rhnsd**, use this command as root:

```
/etc/init.d/rhnsd status
```

On SUSE Linux Enterprise 12 and later, the default time interval is set in **/etc/systemd/system/timers.target.wants/rhnsd.timer**, in this section:

```
[Timer]
OnCalendar=00/4:00
RandomizedDelaySec=30min
```

You can create an overriding drop-in file for **rhnsd.timer** using **systemctl**:


```
systemctl edit rhnsd.timer
```

For example, if you want configure a two hour time interval:

```
[Timer]
OnCalendar=00/2:00
```

The file will be saved as `/etc/systemd/system/rhnsd.timer.d/override.conf`.

For more information about system timers, see the `systemd.timer` and `systemctl` manpages.

Push via SSH

Push via SSH is used in environments where traditional clients cannot reach the Uyuni Server directly. In this environment, clients are located in a firewall-protected zone called a DMZ. No system within the DMZ is authorized to open a connection to the internal network, including the Uyuni Server.

The Push via SSH method creates an encrypted tunnel from the Uyuni Server on the internal network to the clients located on the DMZ. After all actions and events are executed, the tunnel is closed.

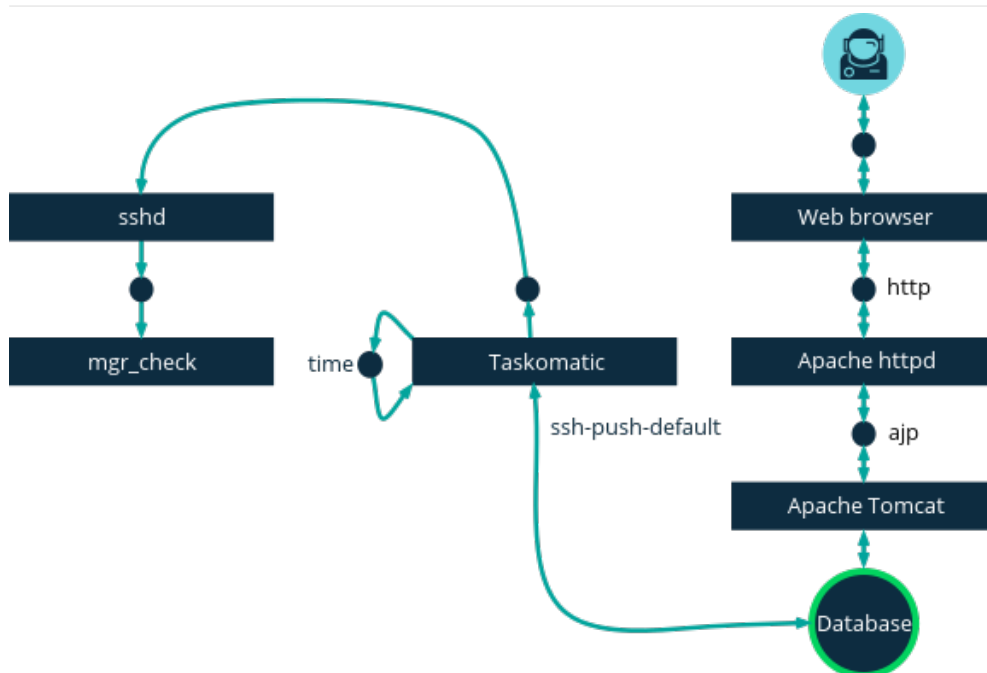
The server uses SSH to contact the clients at regular intervals, checking in and performing scheduled actions and events.

This contact method works for traditional clients only. For Salt clients, use Push via Salt SSH.



Re-installing systems using the provisioning model is not currently supported on clients managed with push via SSH.

This image demonstrates the push via SSH process path. All items left of the `Taskomatic` block represent processes running on a Uyuni client.



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

```
ssh_push_port_http = high_port_1
ssh_push_port_https = high_port_2
```

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

```
ssh_push_use_hostname = true
```

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

```
taskomatic.ssh_push_workers = number
```

For security reasons, you might want to use `sudo` with SSH, to access the system as an unprivileged user instead of as root.

Procedure: Configuring Unprivileged SSH Access

1. Ensure you have the latest `spacewalk-taskomatic` and `spacewalk-certs-tools` packages installed on the Uyuni Server.
2. On each client system, create an appropriate unprivileged user.

- On each client system, open the `/etc/sudoers` file and comment out these lines:

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

- On each client system, in the **User privilege specification** section, add these lines:

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

- On each client system, in the `/home/user/.bashrc` file, add these lines:

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

- On the Uyuni Server, in the `/etc/rhn/rhn.conf` configuration file, add or amend this line to include the unprivileged username:

```
ssh_push_sudo_user = <user>
```

Because clients are in the DMZ and cannot reach the server, you need to use the **mgr-ssh-push-init** tool to register them with the Uyuni Server.

To use the tool, you will need the client hostname or IP address, and the path to a valid bootstrap script on the Uyuni Server. For more information about bootstrapping, see [**Client-configuration** > **Registration-bootstrap** >].

The bootstrap script will need to have an activation key associated with it that is configured for Push via SSH. For more information on activation keys, see [**Client-configuration** > **Activation-keys** >].

Before you begin, you need to ensure that you have specified which ports to use for SSH tunneling. If you have registered clients before changing the port numbers, they will need to be registered again.



Clients that are managed with Push via SSH cannot reach the server directly. When you use the **mgr-ssh-push-init** tool, the **rhn** daemon is disabled.

Procedure: Registering Clients with Push via SSH

- At the command prompt on the Uyuni Server, as root, execute this command:

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

OPTIONAL: You can remove the `--tunnel` option, if you do not want to use tunneling.

2. Verify that the SSH connection is active:

```
# ssh -i /root/.ssh/id_susemanager -R <high_port>:<susemanager>:443 \
<client> zypper ref
```

Example: API Access to Push via SSH

You can use the API to manage which contact method to use. This example Python code sets the contact method to `ssh-push`.

Valid values are:

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

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

If you have a client that has already been registered, and you want to migrate it to use Push via SSH, some extra steps are required. You can use the `mgr-ssh-push-init` tool to set up your client.

Procedure: Migrating Registered Systems to Push via SSH

1. At the command prompt on the Uyuni Server, as root, set up the client:

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

2. Using the Uyuni Web UI, change the client's contact method to `ssh-push` or `ssh-push-tunnel`.
3. OPTIONAL: If you need to edit an existing activation key, you can do so with this command:

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

You can also use Push via SSH for clients that connect using a Uyuni Proxy. Ensure your proxy is updated before you begin.

Procedure: Registering Clients with Push via SSH to a Proxy

1. At the command prompt on the Uyuni Proxy, as root, set up the client:

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

2. At the command prompt on the Uyuni Server, copy the SSH key to the proxy:

```
mgr-ssh-push-init --client <proxy>
```

Push via Salt SSH

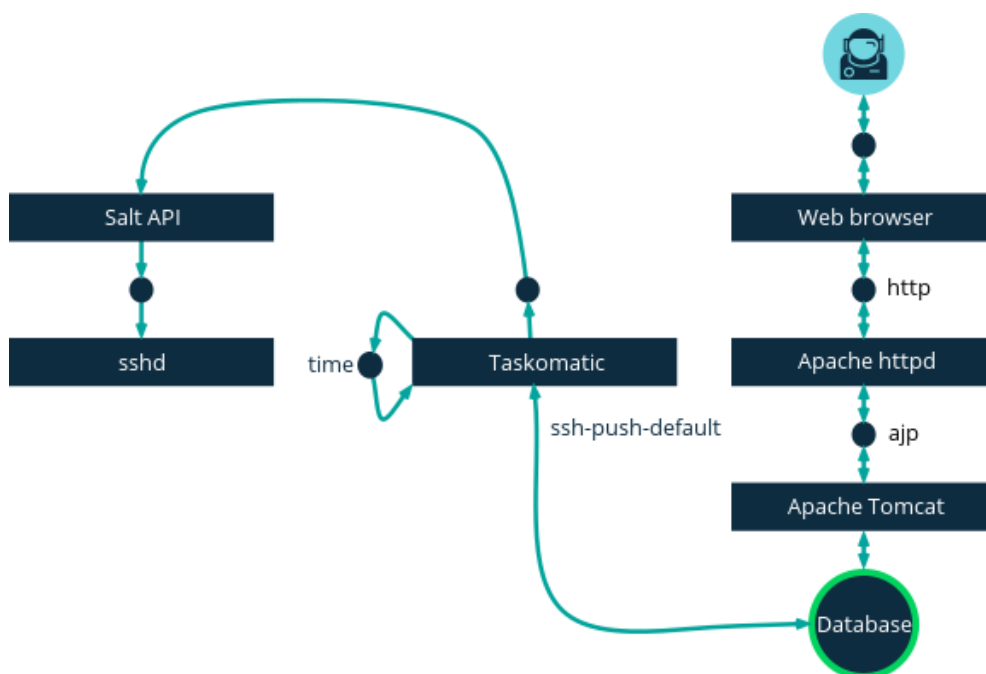
Push via Salt SSH is used in environments where Salt clients cannot reach the Uyuni Server directly. In this environment, clients are located in a firewall-protected zone called a DMZ. No system within the DMZ is authorized to open a connection to the internal network, including the Uyuni Server.

The Push via Salt SSH method creates an encrypted tunnel from the Uyuni Server on the internal network to the clients located on the DMZ. After all actions and events are executed, the tunnel is closed.

The server uses the **salt-ssh** tool to contact the clients at regular intervals, checking in and performing scheduled actions and events. For more information about Salt SSH, see [**Salt > Salt-ssh >**].

This contact method works for Salt clients only. For traditional clients, use Push via SSH.

This image demonstrates the Push via Salt SSH process path. All items left of the **Taskomatic** block represent processes running on a Uyuni client.



To use Push via Salt SSH, you must have the SSH daemon running on the client, and reachable by the **salt-api** daemon running on the Uyuni Server. Additionally, Python must be available on the remote system, and be a version supported by Salt.



Red Hat Enterprise Linux 5, CentOS 5, and earlier are not supported, as they use unsupported versions of Python.

Procedure: Registering Clients with Push via Salt SSH

1. In the Uyuni Web UI, navigate to **Systems > Bootstrapping** and complete the appropriate fields.
2. Select an activation key with the Push via SSH contact method configured. For more information about activation keys, see [**Client-configuration > Activation-keys >**].
3. Check the **Manage system completely via SSH** checkbox.
4. Click [**Bootstrap**] to begin registration.
5. Confirm that the system has been registered correctly by navigating to **Systems > Overview**.

Available Parameters

When you are configuring Push via Salt SSH, you can modify parameters that are used when a system is registered, including the host, activation key, and password. The password is used only for bootstrapping, it is not saved anywhere. All future SSH sessions are authorized via a key/certificate pair. These parameters are configured in **Systems > Bootstrapping**.

You can also configure persistent parameters that are used system-wide, including the sudo user. For more information on configuring the sudo user, see [**Client-configuration > Contact-methods-pushssh >**].

Action Execution

The Push via Salt SSH feature uses taskomatic to execute scheduled actions using **salt-ssh**. The taskomatic job periodically checks for scheduled actions and executes them. Unlike Push via SSH on traditional clients, the Push via Salt SSH feature executes a complete **salt-ssh** call based on the scheduled action.

By default, actions are executed sequentially on each client, and only one action will run at a time. You can increase the number of actions that can be executed in parallel, by adding this line to your configuration file, and adjusting the **number** upwards. We recommend you keep the number of parallel actions low, to avoid problems:

```
taskomatic.com.redhat.rhn.taskomatic.task.MinionActionExecutor.parallel_threads=N
org.quartz.threadPool.threadCount=<number>
```

This will adjust the number of actions that can run in parallel on any one client. If actions need to be run on multiple clients, actions will always be executed sequentially on each client.

If the clients are connected through a Uyuni Proxy, you will need to adjust the **MaxSessions** settings on the proxy. In this case, set the number of parallel connections to be three times the total number of clients.

Future Features

There are some features that are not yet supported on Push via Salt SSH. These features will not work on Salt SSH clients:

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

For more information about Salt SSH, see <https://docs.saltstack.com/en/latest/topics/ssh/>.

OSAD

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



Use OSAD in addition to **rhnsd**. If you disable **rhnsd** your client will be shown as not checking in after 24 hours.

OSAD has several distinct components:

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

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

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

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

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

Procedure: Enabling OSAD

1. At the command prompt on the Uyuni Server, as root, start the **osa-dispatcher** service:

```
systemctl start osa-dispatcher
```

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

```
systemctl start osad
```

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

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

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

Troubleshooting OSAD

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

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

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

```
jabber soft nofile 5100
jabber hard nofile 6000
```

Calculate the limits required for your environment by adding 100 to the number of clients for the soft limit, and 1000 to the current number of clients for the hard limit.

In the example above, we have assumed 500 current clients, so the soft limit is 5100, and the hard limit is 6000.

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

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

Client Registration

Register Clients with the Web UI

Registering clients with the Uyuni Web UI works for Salt clients only.



Do not register a Uyuni Server to itself. The Uyuni Server must be managed individually.

Procedure: Registering Clients with the Web UI

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

When the bootstrap process has completed, your client will be listed at **Systems > System List**.



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



To register and use CentOS 6, Oracle Linux 6, Red Hat Enterprise Linux 6, or SUSE Linux Enterprise Server with Expanded Support 6 clients, you need to configure the Uyuni Server to support older types of SSL encryption. For more information about how to resolve this error, see **Registering Older Clients** at [**Client-configuration > Tshoot-clients >**].

Register Clients with a Bootstrap Script

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

To register clients using a bootstrap script, we recommend you create a template bootstrap script to begin, which can then be copied and modified. The bootstrap script you create is executed on the client when it is registered, and ensures all the necessary packages are deployed to the client. There are some parameters contained in the bootstrap script, which ensure the client system can be assigned to its base channel, including activation keys and GPG keys.

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



A bootstrap repository is needed for non-SLE clients in general and for SLE clients before version 15. A bootstrap offers packages for installing Salt on clients and for registering Salt or traditional clients. For information about creating a bootstrap repository, see [**Client-configuration > Bootstrap-repository >**].

If you are bootstrapping Salt clients using the Web UI, you will need to ensure that the client system has Python installed before you begin. For Salt clients running SUSE Linux Enterprise Server 12 or older, you will also require the `python-xml` package.



GPG Keys and Uyuni Client Tools

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



openSUSE Leap 15 and SLES 15 and Python 3

openSUSE Leap 15 and SLE 15 use Python 3 by default. Bootstrap scripts based on Python 2 must be re-created for openSUSE Leap 15 and SLE 15 systems. Attempting to register openSUSE Leap 15 or SLE 15 systems using Python 2 bootstrap scripts will fail.

Create a Bootstrap Script

You can use the Uyuni Web UI to create an editable bootstrap script.

Procedure: Creating a Bootstrap Script

1. In the Uyuni Web UI, navigate to **Admin > Manager Configuration > Bootstrap Script**.
2. In the **SUSE Manager Configuration - Bootstrap** dialog, uncheck the **Bootstrap using Salt** checkbox if you are installing a traditional client. For Salt clients, leave it checked.

3. The required fields are pre-populated with values derived from previous installation steps. For details on each setting, see [[Reference > Admin >](#)].
4. Click [**Update**] create the script.
5. The bootstrap script is generated and stored on the server in the `/srv/www/htdocs/pub/bootstrap` directory. Alternatively, you can access the bootstrap script over HTTPS. Replace `example.com` with the host name of your Uyuni Server:

```
https://<example.com>/pub/bootstrap/bootstrap.sh
```



Do not disable SSL in your bootstrap script. Ensure that **Enable SSL** is checked in the Web UI, or that the setting `USING_SSL=1` exists in the bootstrap script. If you disable SSL, the registration process requires custom SSL certificates. For more about custom certificates, see [[Administration > Ssl-certs >](#)].



To register and use CentOS 6, Oracle Linux 6, Red Hat Enterprise Linux 6, or SUSE Linux Enterprise Server with Expanded Support 6 clients, you need to configure the Uyuni Server to support older types of SSL encryption. For more information about how to resolve this error, see [Registering Older Clients](#) at [[Client-configuration > Tshoot-clients >](#)].

Editing a Bootstrap Script

You can copy and modify the template bootstrap script you created to customize it. A minimal requirement when modifying a bootstrap script for use with Uyuni is the inclusion of an activation key. Most packages are signed with GPG, so you will also need to have trusted GPG keys on your system to install them.

In this procedure, you will need to know the exact name of your activation keys. Navigate to **Home > Overview** and, in the **Tasks** box, click **Manage Activation Keys**. All keys created for channels are listed on this page. You must enter the full name of the key you wish to use in the bootstrap script exactly as presented in the key field. For more information about activation keys, see [[Client-configuration > Activation-keys >](#)].

Procedure: Modifying a Bootstrap Script

1. On your Uyuni server, as root at the command line change to the bootstrap directory with:

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

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

```
cp bootstrap.sh bootstrap-sles12.sh
cp bootstrap.sh bootstrap-sles15.sh
```

- Open **bootstrap-sles12.sh** for modification. Scroll down until you can see the text shown below. If **exit 1** exists in the file, comment it out by typing a hash or pound sign (#) at the beginning of the line. This activates the script. Enter the name of the key for this script in the **ACTIVATION_KEYS=** field:

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

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

- When you have finished, save the file, and repeat this procedure for the second bootstrap script.

Connect Clients

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

Procedure: Running the Bootstrap Script

- On the Uyuni Server, log in as root. At the command prompt, and change to the bootstrap directory:

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

- Run this command to execute the bootstrap script on the client; replace **EXAMPLE.COM** with the host name of your client:

```
cat bootstrap-sles12.sh | ssh root@EXAMPLE.COM /bin/bash
```

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

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



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

Register on the Command Line (Salt)

Instead of the Web UI, you can use the command line to register a Salt client. This procedure requires that you have installed the Salt package on the Salt client before registration. For SLE 12 based clients, you also must have activated the **Advanced Systems Management** module.



Registering on the command line is also possible with traditional clients, but it requires more steps. It is not covered here. Use the bootstrap script procedure to register traditional clients. For more information, see [registration-bootstrap.pdf](#).

Procedure: Registering Clients Using the Command Line

1. Choose a client configuration file located at:

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

or:

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

This is sometimes also called a minion file.

2. Add the Uyuni Server or Proxy FQDN as the **master**, and the activation key, to the client configuration file:

```
master: SERVER.EXAMPLE.COM
server_id_use_src: adler32
enable_legacy_startup_events: False
enable_fqdns_grains: False
grains:
  susemanager:
    activation_key: "<Activation_Key_Name>"
```

3. Restart the **salt-minion** service:

```
systemctl restart salt-minion
```

4. On the Uyuni Server, accept the new client key; replace **<client>** with the name of your client:

```
salt-key -a '<client>'
```



To register and use CentOS 6, Oracle Linux 6, Red Hat Enterprise Linux 6, or SUSE Linux Enterprise Server with Expanded Support 6 clients, you need to configure the Uyuni Server to support older types of SSL encryption. For more information about how to resolve this error, see [Registering Older Clients](#) at [[Client-configuration](#) > [Tshoot-clients](#) >].

SUSE Client Registration

Registering openSUSE Clients

This section contains information about registering Salt clients running openSUSE operating systems. Uyuni supports openSUSE Leap 15 clients using Salt. Traditional clients are not supported.

Bootstrapping is supported for starting openSUSE clients and performing initial state runs such as setting repositories and performing profile updates.

Add Software Channels

Before you register openSUSE clients to your Uyuni Server, check that you have the openSUSE product enabled, and the required channels are fully synchronized.

The products you need for this procedure are:

Table 11. OpenSUSE Channels - CLI

OS Version	Base Channel	Client Channel	Updates Channel	Other Channels
openSUSE Leap 15.1	opensuse_leap15_1	opensuse_leap15_1-uyuni-client	opensuse_leap15_1-updates	opensuse_leap15_1-non-oss and opensuse_leap15_1-non-oss-updates
openSUSE Leap 15.2	opensuse_leap15_2	opensuse_leap15_2-uyuni-client	opensuse_leap15_2-updates	opensuse_leap15_2-non-oss and opensuse_leap15_2-non-oss-updates

Procedure: Adding Software Channels at the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the `spacewalk-common-channels` command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.

2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

Procedure: Checking Synchronization Progress from the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



openSUSE channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, some operating systems do not trust the GPG key for the Uyuni client tools. The clients can be successfully bootstrapped without the GPG key being trusted. However, you will not be able to install new client tool packages or update them until the keys are trusted.

Procedure: Trusting GPG Keys on Clients

1. On the Uyuni Server, at the command prompt, check the contents of the **/srv/www/htdocs/pub/** directory. This directory contains all available public keys. Take a note of the key that applies to the client you are registering.
2. Open the relevant bootstrap script, locate the **ORG_GPG_KEY=** parameter and add the required key. For example:

```
uyuni-gpg-pubkey-0d20833e.key
```

You do not need to delete any previously stored keys.

3. If you are bootstrapping clients from the Uyuni Web UI, you will need to use a Salt state to trust the key. Create the Salt state and assign it to the organization. You can then use an activation key and configuration channels to deploy the key to the clients.

Register Clients

To register your openSUSE clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

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

For more information on registering your clients, see [[Client-configuration](#) > [Registration-overview](#) >].

Registering SUSE Linux Enterprise Server with Expanded Support Clients

This section contains information about registering traditional and Salt clients running SUSE Linux Enterprise Server with Expanded Support (Expanded Support) operating systems.

Expanded Support clients are based on Red Hat Enterprise Linux or CentOS.

They are sometimes also called SLESES, RES or Red Hat Expanded Support.



You are responsible for arranging access to Red Hat or CentOS base media repositories and installation media.



SUSE does not provide support for Expanded Support systems on Uyuni.



Traditional clients are not available on Expanded Support 8. Expanded Support 8 clients are only supported as Salt clients.

Add Software Channels

For Expanded Support clients, some required packages are contained on the Red Hat Enterprise Linux or CentOS installation media. You must have these packages installed before you can register a Expanded Support client.

The Expanded Support product is provided by SUSE Customer Center. This also includes the client tools package.

Before you register Expanded Support clients to your Uyuni Server, check that you have the corresponding Expanded Support product enabled, and the required channels are fully synchronized.

You need to select two different sets of channels, one for Expanded Support and the other for the Client Tools.

You will need an activation key associated with the correct Expanded Support channels. For more information about activation keys, see [[Client-configuration](#) > [Activation-keys](#) >].

The channels you need for this procedure are:

Table 12. ES Channels - CLI

OS Version	Base Channel	Client Channel	Tools Channel
Expanded Support 6	rhel-x86_64-server-6	-	res6-suse-manager-tools-x86_64

OS Version	Base Channel	Client Channel	Tools Channel
Expanded Support 7	rhel-x86_64-server-7	-	res7-suse-manager-tools-x86_64
Expanded Support 8	rhel-x86_64-server-8	-	res8-suse-manager-tools-x86_64

Procedure: Adding Software Channels at the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the `spacewalk-common-channels` command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```

Add Base Media

The base Expanded Support channel does not contain any packages, because SUSE does not provide Red Hat Enterprise Linux or CentOS base media. You will need to obtain base media from Red Hat or CentOS, which you can add as a child channel to the Expanded Support parent channel. To ensure you have all the packages you need, use a full DVD image, not a minimal or JeOS image.

You can use Uyuni custom channels to set up the Red Hat Enterprise Linux or CentOS media. All packages on the base media are mirrored into a child channel.

The details you need for this procedure are:

Table 13. ES Custom Repository Settings

Option	Parameter
Repository URL	The content URL provided by Red Hat CDN or CentOS
Has Signed Metadata?	Uncheck all Red Hat Enterprise repositories
SSL CA Certificate	<code>redhat-uep</code>
SSL Client Certificate	<code>Entitlement-Cert-date</code>
SSL Client Key	<code>Entitlement-Key-date</code>

Procedure: Creating Custom Repositories

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Repositories**.
2. Click [**Create Repository**] and set the appropriate parameters for the **main** repository.

3. Click **[Create Repository]**.
4. Repeat for all repositories you need to create.

The channels you need for this procedure are:

Table 14. ES Custom Channels

OS Version	Base Channel	Tools Channel
Expanded Support 6	RHEL Expanded Support 6	RES6 SUSE-Manager-Tools x86_64
Expanded Support 7	RHEL Expanded Support 7	RES7 SUSE-Manager-Tools x86_64
Expanded Support 8	RHEL or SLES ES or CentOS 8 Base	RHEL8-Pool for x86_64

When you have created the repositories, you can create the custom channels, one for each repository:

Procedure: Creating Custom Channels

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Channels**.
2. Click **[Create Channel]** and set the appropriate parameters for the channels.
3. In the **Parent Channel** field, select the appropriate base channel.
4. Click **[Create Channel]**.
5. Repeat for all channels you need to create. There should be one custom channel for each custom repository.

You can check that you have created all the appropriate channels and repositories, by navigating to **Software > Channel List > All**.



For Red Hat 8 clients, add both the Base and Appstream channels. You will require packages from both channels. If you do not add both channels, you will not be able to create the bootstrap repository, due to missing packages.

When you have created all the channels, you can associate them with the repositories you created:

Procedure: Associating Channels with Repositories

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Channels**, and click the channel to associate.
2. Navigate to the **Repositories** tab, and check the repository to associate with this channel.
3. Click **[Update Repositories]** to associate the channel and the repository.
4. Repeat for all channels and repositories you need to associate.

5. OPTIONAL: Navigate to the **Sync** tab to set a recurring schedule for synchronization of this repository.
6. Click [**Sync Now**] to begin synchronization immediately.

Procedure: Adding Base Media to Custom Channels

1. On the Uyuni Server, at the command prompt, as root, copy the base media image to the **/tmp/** directory.
2. Create a directory to contain the media content. Replace **<os_name>** with either **sleses6**, **sleses7** or **sleses8**:

```
mkdir -p /srv/www/htdocs/pub/<os_name>
```

3. Mount the image:

```
mount -o loop /tmp/<iso_filename> /srv/www/htdocs/pub/<os_name>
```

When the image is mounted, you can synchronize the base media. To manually synchronize the channels, navigate to **Software > Manage > Channels**. Click each channel in the list, and navigate to menu:[Repositories > Sync]. Click [**Sync Now**] to begin synchronization immediately. You can also create a synchronization schedule from this screen.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

Procedure: Checking Synchronization Progress from the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



The Expanded Support channels can be very large. The initial channel synchronization can sometimes take up to several hours.

When the initial synchronization is complete, we recommended you clone the channel before you work with it. This gives you a backup of the original synchronization data.

Register Expanded Support Clients

Your Expanded Support clients are now ready to be registered.

For more information on registering your clients, see [**Client-configuration** > **Registration-overview** >].



To register and use SUSE Linux Enterprise Server with Expanded Support 6 clients, you need to configure the Uyuni Server to support older types of SSL encryption. For more information about how to resolve this error, see [Registering Older Clients](#) at [**Client-configuration** > **Tshoot-clients** >].

Red Hat Client Registration

Registering Red Hat Enterprise Linux Clients with CDN

If you are running Red Hat Enterprise Linux clients directly, rather than using SUSE Linux Enterprise Server with Expanded Support, you need to use Red Hat sources to retrieve and update packages. This section contains information about using the Red Hat content delivery network (CDN) to register traditional and Salt clients running Red Hat Enterprise Linux operating systems.

For information about using Red Hat update infrastructure (RHUI) instead, see [**Client-configuration > Clients-rh-rhui >**].



Red Hat Enterprise Linux clients are based on Red Hat and are unrelated to SUSE Linux Enterprise Server with Expanded Support, RES, Red Hat, or SUSE Linux Enterprise Server. You are responsible for arranging access to Red Hat base media repositories and RHEL installation media, as well as connecting Uyuni Server to the Red Hat content delivery network. You must obtain support from Red Hat for all your RHEL systems. If you do not do this, you might be violating your terms with Red Hat.



Traditional clients are available on Red Hat Enterprise Linux 6 and 7 only. Red Hat Enterprise Linux 8 clients are supported as Salt clients.

Import Entitlements and Certificates

Red Hat clients require a Red Hat certificate authority (CA) and entitlement certificate, and an entitlement key.

Entitlement certificates are embedded with expiration dates, which match the length of the support subscription. To avoid disruption, you will need to repeat this process at the end of every support subscription period.

Red Hat supply a subscription manager tool to manage subscription assignments. It runs locally to track installed products and subscriptions. Clients must be registered with the subscription manager to obtain certificates.

Red Hat clients use a URL to replicate repositories. The URL will change depending on where the Red Hat client is registered.

Red Hat clients can be registered in three different ways:

- Red Hat content delivery network (CDN) at redhat.com
- Red Hat Satellite Server
- Red Hat update infrastructure (RHUI) in the cloud

This guide covers clients registered to Red Hat CDN. You must have at least one system registered to the CDN, with an authorized subscription for repository content.

For information about using Red Hat update infrastructure (RHUI) instead, see [[Client-configuration > Clients-rh-rhui >](#)].



Satellite certificates for client systems require a Satellite server and subscription. Clients using Satellite certificates are not supported with Uyuni Server.



Entitlement certificates are embedded with expiration dates, which match the length of the support subscription. To avoid disruption, you will need to repeat this process at the end of every support subscription period.

Red Hat supplies the subscription-manager tool to manage subscription assignments. It runs locally on the client system to track installed products and subscriptions. Register to redhat.com with subscription-manager, then follow this procedure to obtain certificates.

Procedure: Registering Clients to Subscription Manager

1. On the client system, at the command prompt, register with the subscription manager tool:

```
subscription-manager register
```

Enter your Red Hat Portal username and password when prompted.

2. Copy your entitlement certificate and key from the client system, to a location that the Uyuni Server can access:

```
cp /etc/pki/entitlement/ /<example>/entitlement/
```



Your entitlement certificate and key will both have a file extension of **.pem**. The key will also have **key** in the filename.

3. Copy the Red Hat CA Certificate file from the client system, to the same web location as the entitlement certificate and key:

```
cp /etc/rhsm/ca/redhat-uep.pem /example/entitlement
```

To manage repositories on your Red Hat client, you need to import the CA and entitlement certificates to the Uyuni Server. This requires that you perform the import procedure three times, to create three entries: one each for the entitlement certificate, the entitlement key, and the Red Hat certificate.

Procedure: Importing Certificates to the Server

1. On the Uyuni Server Web UI, navigate to **Systems > Autoinstallation > GPG and SSL Keys**.
2. Click **[Create Stored Key/Cert]** and set these parameters for the entitlement certificate:
 - In the **Description** field, type **Entitlement-Cert-date**.
 - In the **Type** field, select **SSL**.
 - In the **Select file to upload** field, browse to the location where you saved the entitlement certificate, and select the **.pem** certificate file.
3. Click **[Create Key]**.
4. Click **[Create Stored Key/Cert]** and set these parameters for the entitlement key:
 - In the **Description** field, type **Entitlement-key-date**.
 - In the **Type** field, select **SSL**.
 - In the **Select file to upload** field, browse to the location where you saved the entitlement key, and select the **.pem** key file.
5. Click **[Create Key]**.
6. Click **[Create Stored Key/Cert]** and set these parameters for the Red Hat certificate:
 - In the **Description** field, type **redhat-uep**.
 - In the **Type** field, select **SSL**.
 - In the **Select file to upload** field, browse to the location where you saved the Red Hat certificate, and select the certificate file.
7. Click **[Create Key]**.

Prepare Custom Repositories and Channels

To mirror the software from the Red Hat CDN, you need to create custom channels and repositories in Uyuni that are linked to the CDN by a URL. You must have entitlements to these products in your Red Hat Portal for this to work correctly. You can use the subscription manager tool to get the URLs of the repositories you want to mirror:

```
subscription-manager repos
```

You can use these repository URLs to create custom repositories. This allows you to mirror only the content you need to manage your clients.



You can only create custom versions of Red Hat repositories if you have the correct entitlements in your Red Hat Portal.

The details you need for this procedure are:

Table 15. Red Hat Custom Repository Settings

Option	Setting
Repository URL	The content URL provided by Red Hat CDN
Has Signed Metadata?	Uncheck all Red Hat Enterprise repositories
SSL CA Certificate	redhat-uep
SSL Client Certificate	Entitlement-Cert-date
SSL Client Key	Entitlement-Key-date

Procedure: Creating Custom Repositories

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Repositories**.
2. Click [**Create Repository**] and set the appropriate parameters for the **main** repository.
3. Click [**Create Repository**].
4. Repeat for all repositories you need to create.

The channels you need for this procedure are:

Table 16. Red Hat Custom Channels

OS Version	Base Product	Base Channel
Red Hat 6	RHEL6-Pool for x86_64	rhel6-pool-x86_64
Red Hat 7	RHEL7-Pool for x86_64	rhel7-pool-x86_64
Red Hat 8	RHEL8-Pool for x86_64	rhel8-pool-x86_64

When you have created the repositories, you can create the custom channels, one for each repository:

Procedure: Creating Custom Channels

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Channels**.
2. Click [**Create Channel**] and set the appropriate parameters for the channels.
3. In the **Parent Channel** field, select the appropriate base channel.
4. Click [**Create Channel**].
5. Repeat for all channels you need to create. There should be one custom channel for each custom repository.

You can check that you have created all the appropriate channels and repositories, by navigating to **Software > Channel List > All**.



For Red Hat 8 clients, add both the Base and Appstream channels. You will require packages from both channels. If you do not add both channels, you will not be able to create the bootstrap repository, due to missing packages.

When you have created all the channels, you can associate them with the repositories you created:

Procedure: Associating Channels with Repositories

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Channels**, and click the channel to associate.
2. Navigate to the **Repositories** tab, and check the repository to associate with this channel.
3. Click **[Update Repositories]** to associate the channel and the repository.
4. Repeat for all channels and repositories you need to associate.
5. OPTIONAL: Navigate to the **Sync** tab to set a recurring schedule for synchronization of this repository.
6. Click **[Sync Now]** to begin synchronization immediately.

Add Software Channels

Before you register Red Hat clients to your Uyuni Server, check that you have the Red Hat product enabled, and the required channels are fully synchronized.

The channels you need for this procedure are:

Table 17. Red Hat Channels - CLI

OS Version	Base Channel	Client Channel	Tools Channel
Red Hat 6	rhel-x86_64-server-6	-	res6-suse-manager-tools-x86_64
Red Hat 7	rhel-x86_64-server-7	-	res7-suse-manager-tools-x86_64
Red Hat 8	rhel-x86_64-server-8	-	res8-suse-manager-tools-x86_64

Procedure: Adding Software Channels at the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```



The client tools channel provided by **spacewalk-common-channels** is sourced from Uyuni and not from SUSE.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

Procedure: Checking Synchronization Progress from the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



Red Hat Enterprise Linux channels can be very large. Synchronization can sometimes take several hours.

Procedure: OPTIONAL: Creating a Salt State to Deploy Configuration Files

1. On the Uyuni Server Web UI, navigate to **Configuration > Channels**.
2. Click **[Create State Channel]**.
 - In the **Name** field, type **subscription-manager: disable yum plugins**.
 - In the **Label** field, type **subscription-manager-disable-yum-plugins**.
 - In the **Description** field, type **subscription-manager: disable yum plugins**.
 - In the **SLS Contents** field, leave it empty.
3. Click **[Create Config Channel]**
4. Click **[Create Configuration File]**
 - In the **Filename/Path** field type **/etc/yum/pluginconf.d/subscription-manager.conf**.
 - In the **File Contents** field type:

```
[main]
enabled=0
```

1. Click **[Create Configuration File]**
2. Take note of the value of the field **Salt Filesystem Path`**.
3. Click on the name of the Configuration Channel.
4. Click on **View/Edit 'init.sls' File**
 - In the **File Contents** field, type:

```
configure_subscription-manager-disable-yum-plugins:
  cmd.run:
    - name: subscription-manager config --rhsm.auto_enable_yum_plugins=0
    - watch:
      - file: /etc/yum/pluginconf.d/subscription-manager.conf
  file.managed:
    - name: /etc/yum/pluginconf.d/subscription-manager.conf
    - source: salt:///etc/yum/pluginconf.d/subscription-manager.conf
```

1. Click **[Update Configuration File]**.



The **Creating a Salt State to Deploy Configuration Files** procedure is optional.

Procedure: Creating a System Group for Red Hat Enterprise Linux Clients

1. On the Uyuni Server Web UI, navigate to **Systems > System Groups**.
2. Click **[Create Group]**.
 - In the **Name** field, type **rhel-systems**.
 - In the **Description** field, type **All RHEL systems**.
3. Click **[Create Group]**.
4. Click **States** tab.
5. Click **Configuration Channels** tab.
6. Type **subscription-manager: disable yum plugins** at the search box.
7. Click **[Search]** and the state will appear.
8. Click the checkbox for the state at the **Assign** column.
9. Click **[Save changes]**.
10. Click **[Confirm]**.

If you already have RHEL systems added to Uyuni, assign them to the new system group, and then apply the highstate.

Procedure: Adding the System Group to Activation Keys

You need to modify the activation keys you used for RHEL systems to include the system group created above.

1. On the Uyuni Server Web UI, navigate to **Systems > Activation Keys**.
2. For each the Activation Keys you used for RHEL systems, click on it and:
3. Navigate to the **Groups** tab, and the **Join** subtab.
4. Check **Select rhel-systems**.
5. Click [**Join Selected Groups**].

Trust GPG Keys on Clients

By default, some operating systems do not trust the GPG key for the Uyuni client tools. The clients can be successfully bootstrapped without the GPG key being trusted. However, you will not be able to install new client tool packages or update them until the keys are trusted.

Procedure: Trusting GPG Keys on Clients

1. On the Uyuni Server, at the command prompt, check the contents of the `/srv/www/htdocs/pub/` directory. This directory contains all available public keys. Take a note of the key that applies to the client you are registering.
2. Open the relevant bootstrap script, locate the `ORG_GPG_KEY=` parameter and add the required key. For example:

```
uyuni-gpg-pubkey-0d20833e.key
```

You do not need to delete any previously stored keys.

3. If you are bootstrapping clients from the Uyuni Web UI, you will need to use a Salt state to trust the key. Create the Salt state and assign it to the organization. You can then use an activation key and configuration channels to deploy the key to the clients.

Register Clients

To register your Red Hat clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

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

For more information on registering your clients, see [**Client-configuration > Registration-overview >**].



To register and use Red Hat Enterprise Linux 6 clients, you need to configure the Uyuni Server to support older types of SSL encryption. For more information about how to resolve this error, see **Registering Older Clients** at [**Client-configuration > Tshoot-clients >**].

Package Management and Red Hat Enterprise Linux 8 Clients

If you are using Red Hat Enterprise Linux 8 clients, you cannot perform package operations such as installing or upgrading directly from modular repositories like the Red Hat Enterprise Linux Appstream repository. You can use the Appstream filter with content lifecycle management to transform modular repositories into regular repositories.

For more information about content lifecycle management, see [[Administration > Content-lifecycle >](#)].

Registering Red Hat Enterprise Linux Clients with RHUI

If you are running Red Hat Enterprise Linux clients directly, rather than using SUSE Linux Enterprise Server with Expanded Support, you need to use Red Hat sources to retrieve and update packages. This section contains information about using Red Hat update infrastructure (RHUI) to register traditional and Salt clients running Red Hat Enterprise Linux operating systems. If you are running your clients in a public cloud, such as Amazon EC2, use this method.

It is possible to use RHUI in conjunction with the Red Hat content delivery network (CDN) to manage your Red Hat Enterprise Linux subscriptions. For information about using Red Hat CDN, see [[Client-configuration > Clients-rh-cdn >](#)].



Red Hat Enterprise Linux clients are based on Red Hat and are unrelated to SUSE Linux Enterprise Server with Expanded Support, RES, or SUSE Linux Enterprise Server. You are responsible for connecting Uyuni Server to the Red Hat update infrastructure. All clients that get updates using this RHUI certificate need to be correctly licensed, please check with your cloud provider and the Red Hat terms of service for more information.



When Red Hat Enterprise Linux clients registered with RHUI are switched off, Red Hat might declare the certificate invalid. In this case, you need to turn the client on again, or get a new RHUI certificate.



Traditional clients are available on Red Hat Enterprise Linux 6 and 7 only. Red Hat Enterprise Linux 8 clients are supported as Salt clients.

Import Entitlements and Certificates

Red Hat clients require a Red Hat certificate authority (CA) and entitlement certificate, and an entitlement key.

Red Hat clients use a URL to replicate repositories. The URL will change depending on where the Red Hat client is registered.

Red Hat clients can be registered in three different ways:

- Red Hat content delivery network (CDN) at redhat.com
- Red Hat Satellite Server
- Red Hat update infrastructure (RHUI) in the cloud

This guide covers clients registered to Red Hat update infrastructure (RHUI). You must have at least one system registered to RHUI, with an authorized subscription for repository content.

For information about using Red Hat content delivery network (CDN) instead, see [**Client-configuration** › **Clients-rh-cdn** ›].



Satellite certificates for client systems require a Satellite server and subscription. Clients using Satellite certificates are not supported with Uyuni Server.

The entitlement certificates and keys need to be copied from the client system to a location that the Uyuni Server can access.

Your entitlement certificate and the Red Hat CA Certificate file have file extensions of **.crt**. The key has a file extension of **.key**.

Procedure: Copying Certificates to the Server

1. Copy your entitlement certificate and key from the client system, to a location that the Uyuni Server can access:

```
cp /etc/pki/rhui/product/content-<version>.crt /<example>/entitlement/
cp /etc/pki/rhui/content-<version>.key /<example>/entitlement/
```

2. Copy the Red Hat CA Certificate file from the client system, to the same location as the entitlement certificate and key:

```
cp /etc/pki/rhui/cdn.redhat.com-chain.crt /<example>/entitlement
```

To manage repositories on your Red Hat client, you need to import the CA and entitlement certificates to the Uyuni Server. This requires that you perform the import procedure three times, to create three entries: one each for the entitlement certificate, the entitlement key, and the Red Hat certificate.

Procedure: Importing Certificates to the Server

1. On the Uyuni Server Web UI, navigate to **Systems** › **Autoinstallation** › **GPG and SSL Keys**.
2. Click [**Create Stored Key/Cert**] and set these parameters for the entitlement certificate:
 - In the **Description** field, type **Entitlement-Cert-Date**.
 - In the **Type** field, select **SSL**.
 - In the **Select file to upload** field, browse to the location where you saved the entitlement certificate, and select the **.crt** certificate file.

3. Click **[Create Key]**.
4. Click **[Create Stored Key/Cert]** and set these parameters for the entitlement key:
 - In the **Description** field, type **Entitlement-Key-Date**.
 - In the **Type** field, select **SSL**.
 - In the **Select file to upload** field, browse to the location where you saved the entitlement key, and select the **.key** key file.
5. Click **[Create Key]**.
6. Click **[Create Stored Key/Cert]** and set these parameters for the Red Hat certificate:
 - In the **Description** field, type **redhat-cert**.
 - In the **Type** field, select **SSL**.
 - In the **Select file to upload** field, browse to the location where you saved the Red Hat certificate, and select the certificate file.
7. Click **[Create Key]**.

Prepare Custom Repositories and Channels

To mirror the software from RHUI, you need to create custom channels and repositories in Uyuni that are linked to RHUI by a URL. You must have entitlements to these products in your Red Hat Portal for this to work correctly. You can use the `yum` utility to get the URLs of the repositories you want to mirror:

```
yum repolist -v | grep baseurl
```

You can use these repository URLs to create custom repositories. This allows you to mirror only the content you need to manage your clients.



You can only create custom versions of Red Hat repositories if you have the correct entitlements in your Red Hat Portal.

The details you need for this procedure are:

Table 18. Red Hat Custom Repository Settings

Option	Setting
Repository URL	The content URL provided by RHUI
Has Signed Metadata?	Uncheck all Red Hat Enterprise repositories
SSL CA Certificate	redhat-cert
SSL Client Certificate	Entitlement-Cert-Date

Option	Setting
SSL Client Key	Entitlement-Key-Date

Procedure: Creating Custom Repositories

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Repositories**.
2. Click **[Create Repository]** and set the appropriate parameters for the **main** repository.
3. Click **[Create Repository]**.
4. Repeat for all repositories you need to create.

The channels you need for this procedure are:

Table 19. Red Hat Custom Channels

OS Version	Base Product	Base Channel
Red Hat 6	RHEL6-Pool for x86_64	rhel6-pool-x86_64
Red Hat 7	RHEL7-Pool for x86_64	rhel7-pool-x86_64
Red Hat 8	RHEL8-Pool for x86_64	rhel8-pool-x86_64

When you have created the repositories, you can create the custom channels, one for each repository:

Procedure: Creating Custom Channels

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Channels**.
2. Click **[Create Channel]** and set the appropriate parameters for the channels.
3. In the **Parent Channel** field, select the appropriate base channel.
4. Click **[Create Channel]**.
5. Repeat for all channels you need to create. There should be one custom channel for each custom repository.

You can check that you have created all the appropriate channels and repositories, by navigating to **Software > Channel List > All**.



For Red Hat 8 clients, add both the Base and Appstream channels. You will require packages from both channels. If you do not add both channels, you will not be able to create the bootstrap repository, due to missing packages.

When you have created all the channels, you can associate them with the repositories you created:

Procedure: Associating Channels with Repositories

1. On the Uyuni Server Web UI, navigate to **Software > Manage > Channels**, and click the channel to associate.

2. Navigate to the **Repositories** tab, and check the repository to associate with this channel.
3. Click **[Update Repositories]** to associate the channel and the repository.
4. Repeat for all channels and repositories you need to associate.
5. OPTIONAL: Navigate to the **Sync** tab to set a recurring schedule for synchronization of this repository.
6. Click **[Sync Now]** to begin synchronization immediately.

Add Software Channels

Before you register Red Hat clients to your Uyuni Server, check that you have the Red Hat product enabled, and the required channels are fully synchronized.

The channels you need for this procedure are:

Table 20. Red Hat Channels - CLI

OS Version	Base Channel	Client Channel	Tools Channel
Red Hat 6	rhel-x86_64-server-6	-	res6-suse-manager-tools-x86_64
Red Hat 7	rhel-x86_64-server-7	-	res7-suse-manager-tools-x86_64
Red Hat 8	rhel-x86_64-server-8	-	res8-suse-manager-tools-x86_64

Procedure: Adding Software Channels at the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```



The client tools channel provided by **spacewalk-common-channels** is sourced from Uyuni and not from SUSE.

To use RHUI, you need to manually add the required HTTP headers to the configuration file. Without them, you cannot successfully perform a client synchronization.

Procedure: Adding HTTP Headers to the Configuration File

1. Locate the **X-RHUI-ID** and **X-RHUI-SIGNATURE** HTTP headers from your RHUI instance. You

can use these commands on the Red Hat client to get the values from the cloud instance metadata API at **169.254.169.254**:

```
echo "X-RHUI-ID=$(curl -s http://169.254.169.254/latest/dynamic/instance-identity/document|base64|tr -d '\n')"
```

```
echo "X-RHUI-SIGNATURE=$(curl -s http://169.254.169.254/latest/dynamic/instance-identity/signature|base64|tr -d '\n')"
```

2. Open the **/etc/rhn/spacewalk-repo-sync/extra_headers.conf** configuration file, and add or edit these lines with the correct information:

```
[channel_label]
X-RHUI-ID=value
X-RHUI-SIGNATURE=value
```

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

Procedure: Checking Synchronization Progress from the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



Red Hat Enterprise Linux channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, some operating systems do not trust the GPG key for the Uyuni client tools. The clients can be successfully bootstrapped without the GPG key being trusted. However, you will not be able to install new client tool packages or update them until the keys are trusted.

Procedure: Trusting GPG Keys on Clients

1. On the Uyuni Server, at the command prompt, check the contents of the **/srv/www/htdocs/pub/** directory. This directory contains all available public keys. Take a note of the key that applies to the client you are registering.

2. Open the relevant bootstrap script, locate the `ORG_GPG_KEY=` parameter and add the required key. For example:

```
uyuni-gpg-pubkey-0d20833e.key
```

You do not need to delete any previously stored keys.

3. If you are bootstrapping clients from the Uyuni Web UI, you will need to use a Salt state to trust the key. Create the Salt state and assign it to the organization. You can then use an activation key and configuration channels to deploy the key to the clients.

Register Clients

To register your Red Hat clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

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

For more information on registering your clients, see [[Client-configuration > Registration-overview >](#)].



To register and use Red Hat Enterprise Linux 6 clients, you need to configure the Uyuni Server to support older types of SSL encryption. For more information, see [Registering Older Clients](#) at [[Client-configuration > Tshoot-clients >](#)].

Package Management and Red Hat Enterprise Linux 8 Clients

If you are using Red Hat Enterprise Linux 8 clients, you cannot perform package operations such as installing or upgrading directly from modular repositories like the Red Hat Enterprise Linux Appstream repository. You can use the Appstream filter with content lifecycle management to transform modular repositories into regular repositories.

For more information about content lifecycle management, see [[Administration > Content-lifecycle >](#)].

CentOS Client Registration

Registering CentOS Clients

This section contains information about registering traditional and Salt clients running CentOS operating systems.



CentOS clients are based on CentOS and are unrelated to SUSE Linux Enterprise Server with Expanded Support, RES, Red Hat, or Expanded Support. You are responsible for arranging access to CentOS base media repositories and CentOS installation media, as well as connecting Uyuni Server to the CentOS content delivery network.



Traditional clients are not available on CentOS 8. CentOS 8 clients are only supported as Salt clients.

Add Software Channels

Before you register CentOS clients to your Uyuni Server, check that you have the CentOS product enabled, and the required channels are fully synchronized.

The channels you need for this procedure are:

Table 21. CentOS Channels - CLI

OS Version	Base Channel	Client Channel	Updates Channel
CentOS 6	centos6	centos6-uyuni-client	centos6-updates
CentOS 7	centos7	centos7-uyuni-client	centos7-updates
CentOS 8	centos8	centos8-uyuni-client	centos8-appstream

Procedure: Adding Software Channels at the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```



The client tools channel provided by **spacewalk-common-channels** is sourced from Uyuni and not from SUSE.



For CentOS 8 clients, add both the Base and Appstream channels. You will require packages from both channels. If you do not add both channels, you will not be able to create the bootstrap repository, due to missing packages.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

Procedure: Checking Synchronization Progress from the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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

Create an Activation Key

You will need to create an activation key that is associated with your CentOS channels.

For more information on activation keys, see [**Client-configuration > Activation-keys >**].

Trust GPG Keys on Clients

By default, some operating systems do not trust the GPG key for the Uyuni client tools. The clients can be successfully bootstrapped without the GPG key being trusted. However, you will not be able to install new client tool packages or update them until the keys are trusted.

Procedure: Trusting GPG Keys on Clients

1. On the Uyuni Server, at the command prompt, check the contents of the **/srv/www/htdocs/pub/** directory. This directory contains all available public keys. Take a note of the key that applies to the client you are registering.
2. Open the relevant bootstrap script, locate the **ORG_GPG_KEY=** parameter and add the required key. For example:

```
uyuni-gpg-pubkey-0d20833e.key
```

You do not need to delete any previously stored keys.

3. If you are bootstrapping clients from the Uyuni Web UI, you will need to use a Salt state to trust the key. Create the Salt state and assign it to the organization. You can then use an activation key and configuration channels to deploy the key to the clients.

Register Clients

CentOS clients are registered in the same way as all other clients. For more information, see [**Client-configuration** > **Registration-overview** >].



To register and use CentOS 6 clients, you need to configure the Uyuni Server to support older types of SSL encryption. For more information about how to resolve this error, see [Registering Older Clients](#) at [**Client-configuration** > **Tshoot-clients** >].

Manage Errata

When you update CentOS clients, the packages do not include metadata about the updates. You can use a third-party errata service to obtain this information.

Procedure: Installing an Errata Service

1. On the Uyuni Server, from the command prompt, as root, add the **sle-module-development-tools** module:

```
SUSEConnect --product sle-module-development-tools/15.2/x86_64
```

2. Install the errata service:

```
zypper in perl-Text-Unidecode
```

3. Create a file for your errata script:

```
touch /usr/local/bin/cent-errata.sh
```

4. Edit the new file to include this script, editing the repository details as required. This script fetches the errata details from an external errata service, unpacks it, and publishes the details:


```
#!/bin/bash
mkdir -p /usr/local/centos
cd /usr/local/centos
rm *.xml
wget -c http://cefs.steve-meier.de/errata.latest.xml
#wget -c https://www.redhat.com/security/data/oval/com.redhat.rhsa-all.xml
wget -c https://www.redhat.com/security/data/oval/com.redhat.rhsa-RHEL7.xml
wget -c http://cefs.steve-meier.de/errata-import.tar
tar xvf errata-import.tar
chmod +x /usr/local/centos/errata-import.pl
export SPACEWALK_USER='<adminname>';export SPACEWALK_PASS='<password>'
/usr/local/centos/errata-import.pl --server '<servername>' \
--errata /usr/local/centos/errata.latest.xml \
--include-channels=centos7-x86_64-updates,centos7-x86_64,centos7-x86_64-extras \
--publish --rhsa-oval /usr/local/centos/com.redhat.rhsa-RHEL7.xml
```

5. Set up a cron job to run the script daily:

```
ln -s /usr/local/bin/cent-errata.sh /etc/cron.daily
```

For more information on this tool, see <https://cefs.steve-meier.de/>.

Oracle Client Registration

Registering Oracle Linux Clients

This section contains information about registering traditional and Salt clients running Oracle Linux operating systems.



Traditional clients are not available on Oracle Linux 8. Oracle Linux 8 clients are only supported as Salt clients.

Add Software Channels

Before you register Oracle Linux clients to your Uyuni Server, check that you have the Oracle Linux product enabled, and the required channels are fully synchronized.

The channels you need for this procedure are:

Table 22. Oracle Channels - CLI

OS Version	Base Channel	Client Channel	Updates Channel
Oracle Linux 6	oraclelinux6	oraclelinux6-uyuni-client	-
Oracle Linux 7	oraclelinux7	oraclelinux7-uyuni-client	-
Oracle Linux 8	oraclelinux8	oraclelinux8-uyuni-client	oraclelinux8-appstream

Procedure: Adding Software Channels at the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```



The client tools channel provided by **spacewalk-common-channels** is sourced from Uyuni and not from SUSE.



For Oracle Linux 8 clients, add both the Base and Appstream channels. You will require packages from both channels. If you do not add both channels, you will not be able to create the bootstrap repository, due to missing packages.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

Procedure: Checking Synchronization Progress from the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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

Create an Activation Key

You will need to create an activation key that is associated with your Oracle Linux channels.

For more information on activation keys, see [**Client-configuration > Activation-keys >**].

Trust GPG Keys on Clients

By default, some operating systems do not trust the GPG key for the Uyuni client tools. The clients can be successfully bootstrapped without the GPG key being trusted. However, you will not be able to install new client tool packages or update them until the keys are trusted.

Procedure: Trusting GPG Keys on Clients

1. On the Uyuni Server, at the command prompt, check the contents of the **/srv/www/htdocs/pub/** directory. This directory contains all available public keys. Take a note of the key that applies to the client you are registering.
2. Open the relevant bootstrap script, locate the **ORG_GPG_KEY=** parameter and add the required key. For example:

```
uyuni-gpg-pubkey-0d20833e.key
```

You do not need to delete any previously stored keys.

3. If you are bootstrapping clients from the Uyuni Web UI, you will need to use a Salt state to trust the key. Create the Salt state and assign it to the organization. You can then use an activation key and configuration channels to deploy the key to the clients.

Register Clients

Oracle Linux clients are registered in the same way as all other clients. For more information, see [**Client-configuration** > **Registration-overview** >].



To register and use Oracle Linux 6 clients, you need to configure the Uyuni Server to support older types of SSL encryption. For more information about how to resolve this error, see [Registering Older Clients](#) at [**Client-configuration** > **Tshoot-clients** >].

Ubuntu Client Registration

Registering Ubuntu 20.04 Clients

This section contains information about registering Salt clients running Ubuntu 20.04 LTS operating systems.

Traditional clients are not supported.



Canonical does not endorse or support Uyuni.

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

Add Software Channels

Before you register Ubuntu clients to your Uyuni Server, check that you have the Ubuntu product enabled, and the required channels are fully synchronized.

The channels you need for this procedure are:

Table 23. Ubuntu Channels - CLI

OS Version	Base Channel	Base Uyuni Channel	Updates Channel	Security Channel	Universe Uyuni Channel	Universe Updates Channel	Client Channel
Ubuntu 20.04	ubuntu-2004-amd64-main for amd64	ubuntu-2004-amd64-main-uyuni	ubuntu-2004-amd64-main-updates-uyuni	ubuntu-2004-amd64-main-security-uyuni	ubuntu-2004-amd64-universe-uyuni	ubuntu-2004-amd64-universe-updates-uyuni	ubuntu-2004-amd64-uyuni-client

Procedure: Adding Software Channels at the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```



You need all the new channels fully synchronized, including Universe (Universe contains important dependencies for Salt), before bootstrapping any Ubuntu client.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

Procedure: Checking Synchronization Progress from the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



Ubuntu channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, some operating systems do not trust the GPG key for the Uyuni client tools. The clients can be successfully bootstrapped without the GPG key being trusted. However, you will not be able to install new client tool packages or update them until the keys are trusted.

Procedure: Trusting GPG Keys on Clients

1. On the Uyuni Server, at the command prompt, check the contents of the **/srv/www/htdocs/pub/** directory. This directory contains all available public keys. Take a note of the key that applies to the client you are registering.
2. Open the relevant bootstrap script, locate the **ORG_GPG_KEY=** parameter and add the required key. For example:

```
uyuni-gpg-pubkey-0d20833e.key
```

You do not need to delete any previously stored keys.

3. If you are bootstrapping clients from the Uyuni Web UI, you will need to use a Salt state to trust the key. Create the Salt state and assign it to the organization. You can then use an activation key and configuration channels to deploy the key to the clients.

Root Access

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

Procedure: Granting Root User Access

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

```
sudo visudo
```

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

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



This procedure grants root access without requiring a password, which is required for registering the client. When the client is successfully installed it will run with root privileges, so the access is no longer required. We recommend that you remove the line from the `sudoers` file after the client has been successfully installed.

Register Clients

To register your Ubuntu clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

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

For more information on registering your clients, see [[Client-configuration](#) > [Registration-overview](#) >].

Registering Ubuntu 16.04 and 18.04 Clients

This section contains information about registering Salt clients running Ubuntu 16.04 LTS, 18.04 LTS operating systems.

Uyuni supports Ubuntu 16.04 LTS and 18.04 LTS clients using Salt.

Traditional clients are not supported.



Canonical does not endorse or support Uyuni.

Bootstrapping is supported for starting Ubuntu clients and performing initial state runs such as setting repositories and performing profile updates. However, the root user on Ubuntu is disabled by default, so

to use bootstrapping, you will require an existing user with **sudo** privileges for Python.

Add Software Channels

Before you register Ubuntu clients to your Uyuni Server, check that you have the Ubuntu product enabled, and the required channels are fully synchronized.

The channels you need for this procedure are:

Table 24. Ubuntu Channels - CLI

OS Version	Base Channel	Base Uyuni Channel	Updates Channel	Security Channel	Universe Uyuni Channel	Universe Updates Channel	Client Channel
Ubuntu 16.04	ubuntu-1604-pool-amd64-uyuni	ubuntu-1604-amd64-main-uyuni	ubuntu-1604-amd64-main-updates-uyuni	ubuntu-1604-amd64-main-security-uyuni	ubuntu-1604-amd64-universe-uyuni	ubuntu-1604-amd64-universe-updates-uyuni	ubuntu-1604-amd64-uyuni-client
Ubuntu 18.04	ubuntu-1804-pool-amd64-uyuni	ubuntu-1804-amd64-main-uyuni	ubuntu-1804-amd64-main-updates-uyuni	ubuntu-1804-amd64-main-security-uyuni	ubuntu-1804-amd64-universe-uyuni	ubuntu-1804-amd64-universe-updates-uyuni	ubuntu-1804-amd64-uyuni-client

Procedure: Adding Software Channels at the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **spacewalk-common-channels** command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```



You need all the new channels fully synchronized, including Universe (Universe contains important dependencies for Salt), before bootstrapping any Ubuntu client.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated

to the repository.

2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

Procedure: Checking Synchronization Progress from the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the **tail** command to check the synchronization log file:

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

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



Ubuntu channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, some operating systems do not trust the GPG key for the Uyuni client tools. The clients can be successfully bootstrapped without the GPG key being trusted. However, you will not be able to install new client tool packages or update them until the keys are trusted.

Procedure: Trusting GPG Keys on Clients

1. On the Uyuni Server, at the command prompt, check the contents of the **/srv/www/htdocs/pub/** directory. This directory contains all available public keys. Take a note of the key that applies to the client you are registering.
2. Open the relevant bootstrap script, locate the **ORG_GPG_KEY=** parameter and add the required key. For example:

```
uyuni-gpg-pubkey-0d20833e.key
```

You do not need to delete any previously stored keys.

3. If you are bootstrapping clients from the Uyuni Web UI, you will need to use a Salt state to trust the key. Create the Salt state and assign it to the organization. You can then use an activation key and configuration channels to deploy the key to the clients.

Root Access

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

Procedure: Granting Root User Access

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

```
sudo visudo
```

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

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



This procedure grants root access without requiring a password, which is required for registering the client. When the client is successfully installed it will run with root privileges, so the access is no longer required. We recommend that you remove the line from the **sudoers** file after the client has been successfully installed.

Register Clients

To register your Ubuntu clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

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

For more information on registering your clients, see [[Client-configuration](#) > [Registration-overview](#) >].

Registering Debian Clients

This section contains information about registering Salt clients running Debian operating systems.



SUSE does not provide support for Debian operating systems. Uyuni allows you to manage Debian clients, but support is not provided. Using Uyuni to manage Debian clients is experimental. These instructions have been tested on Debian 9 Stretch and Debian 10 Buster. Do not rely on Debian clients in a production environment.

Bootstrapping can be used with Debian clients for performing initial state runs, and for profile updates.

Prepare to Register

Some preparation is required before you can register Debian clients to the Uyuni Server:

- If you are using Debian 9, install the **apt-transport-https** package on the client before you attempt to register. On the client, at the command prompt, as root, run:

```
apt install apt-transport-https
```

- Ensure DNS is correctly configured and provides an entry for the client. Alternatively, you can configure the `/etc/hosts` files on both the Uyuni Server and the client with the appropriate entries.
- The client must have the date and time synchronized correctly with the Uyuni Server before registration.

Add Software Channels

The channels you need for this procedure are:

Table 25. Debian Channels - CLI

OS Version	Base Channel	Client Channel	Updates Channel	Security Channel
Debian 9	debian-9-pool-amd64-uyuni	debian-9-amd64-uyuni-client	debian-9-amd64-main-updates-uyuni	debian-9-amd64-main-security-uyuni
Debian 10	debian-10-pool-amd64-uyuni	debian-10-amd64-uyuni-client	debian-10-amd64-main-updates-uyuni	debian-10-amd64-main-security-uyuni

Procedure: Adding Software Channels at the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the `spacewalk-common-channels` command to add the appropriate channels:

```
spacewalk-common-channels \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni Web UI, navigate to **Software > Manage > Channels**, then click the channel associated to the repository.
2. Navigate to the **Repositories** tab, then click **Sync** and check **Sync Status**.

Procedure: Checking Synchronization Progress from the Command Prompt

1. At the command prompt on the Uyuni Server, as root, use the `tail` command to check the synchronization log file:

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

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



Debian channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, some operating systems do not trust the GPG key for the Uyuni client tools. The clients can be successfully bootstrapped without the GPG key being trusted. However, you will not be able to install new client tool packages or update them until the keys are trusted.

Procedure: Trusting GPG Keys on Clients

1. On the Uyuni Server, at the command prompt, check the contents of the `/srv/www/htdocs/pub/` directory. This directory contains all available public keys. Take a note of the key that applies to the client you are registering.
2. Open the relevant bootstrap script, locate the `ORG_GPG_KEY=` parameter and add the required key. For example:

```
uyuni-gpg-pubkey-0d20833e.key
```

You do not need to delete any previously stored keys.

3. If you are bootstrapping clients from the Uyuni Web UI, you will need to use a Salt state to trust the key. Create the Salt state and assign it to the organization. You can then use an activation key and configuration channels to deploy the key to the clients.

Register Clients

To register your Debian clients, you need a bootstrap repository. By default, bootstrap repositories are regenerated daily. You can manually create the bootstrap repository from the command prompt, using this command:

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

For Debian 10, select `debian10-amd64-uyuni` when prompted.

For more information on registering your clients, see [[Client-configuration](#) > [Registration-overview](#) >].

Client Registration on a Proxy

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

These sections contain information on registering Salt clients on a proxy using the Web UI, commands on the command line, or a bootstrap script. There is also information on registering traditional clients using a bootstrap script.

Within the Web UI, proxy pages will show information about both Salt and traditional clients. You can see a list of clients that are connected to a proxy by clicking the name of the proxy in **Systems > System List > Proxy**, then select the **Proxy** subtab of the **Details** tab.

A list of chained proxies for a Salt client can be seen by clicking the name of the client in **Systems > All**, then select the **Connection** subtab of the **Details** tab.

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

Register Clients to a Proxy in the Web UI

You can register Salt clients to a Uyuni Proxy using the Web UI.



A bootstrap repository is needed for non-SLE clients in general and for SLE clients before version 15. A bootstrap offers packages for installing Salt on clients and for registering Salt or traditional clients. For information about creating a bootstrap repository, see [**Client-configuration > Bootstrap-repository >**].

Procedure: Registering Clients to a Proxy in the Web UI

1. In the Uyuni Web UI, navigate to **Systems > Bootstrapping**.
2. In the **Host** field, type the fully-qualified domain name (FQDN) of the client to be bootstrapped.
3. In the **SSH Port** field, type the SSH port number that will be used to connect and bootstrap the client. By default, the SSH port is **22**.
4. In the **User** field, type the username to log in to the client. By default, the username is **root**.
5. In the **Authentication Method** field, select the authentication method to use for bootstrapping the client.
 - For password authentication, in the **Password** field, type password to log in to the client.
 - For SSH Private key authentication, enter the private key and the associated passphrase. The key is only stored for as long as the bootstrapping process takes to complete.
6. In the **Activation Key** field, select the activation key that is associated with the software channel you want to use to bootstrap the client.

7. In the **Proxy** field, select the proxy server you want to register to.
8. By default, the **Disable SSH Strict Key Host Checking** checkbox is selected. This allows the bootstrap process to automatically accept SSH host keys without requiring you to manually authenticate.
9. OPTIONAL: Check the **Manage System Completely via SSH** checkbox. If you check this option, the client will be configured to use SSH for its connection to the server, and no other connection method will be configured.
10. Click **[Bootstrap]** to begin registration.

When the bootstrap process has completed, your client will be listed at **Systems > System List**.

Register on the Command Line (Salt)

Instead of the Web UI, you can use the command line to register a Salt client to a proxy. This procedure requires that you have installed the Salt package on the Salt client before registration. For SLE 12 based clients, you also must have activated the **Advanced Systems Management** module.



Registering traditional clients on the command line is also possible, but it requires more steps. It is not covered here. Use the bootstrap script procedure to register traditional clients. For more information, see [client-proxy-script.pdf](#).

Procedure: Registering Clients to a Proxy Using the Command Line

1. Choose a client configuration file located at:

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

or:

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

This is sometimes also called a minion file.

2. Add the proxy FQDN as the **master** to the client configuration file:

```
master: PROXY123.EXAMPLE.COM
```

3. Restart the **salt-minion** service:

```
systemctl restart salt-minion
```

4. On the server, accept the new client key; replace **<client>** with the name of your client:

```
salt-key -a '<client>'
```

Registering with a Bootstrap Script (Salt and Traditional)

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

Procedure: Registering Clients to a Proxy with a Bootstrap Script

1. Create a client activation key on the Uyuni server using the Web UI. For more information, see [**Client-configuration > Activation-keys >**].
2. On the proxy, execute the **mgr-bootstrap** command line tool as root. If needed, use the additional command line switches to tune your bootstrap script. To install a traditional client instead of a Salt client, ensure you use the **--traditional** switch.

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

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

3. Optional: edit the resulting bootstrap script.
4. Execute the bootstrap script on the clients.

Automate Client Installation

AutoYaST and Kickstart configuration files allow you to automate client system installations. This is useful if you need to install a large number of clients.

For SUSE Linux Enterprise clients, use AutoYaST. When you have created an AutoYaST file, you can upload and manage it using the Uyuni Web UI.

For Red Hat Enterprise Linux clients, use Kickstart. Kickstart files are created, modified, and managed within the Uyuni Web UI.

We recommend that you use PXE boot for installing clients. PXE booting requires a DHCP server that points to your Uyuni Server. The Uyuni Server then acts as a TFTP server.

The TFTP environment is generated with Cobbler. Cobbler can also generate a bootable ISO image. The ISO image can be used to install machines when PXE boot is not an option; for more information, see [**Client-configuration > Cobbler >**].

Preparation

A configured distribution and an autoinstallation profile is required.

Procedure: Preparing a Distribution

1. Provide the files required to start an installation. Unpack an installation medium such as a DVD image on your Server. It contains the Linux kernel, an initrd, and other files required to boot the OS in installation mode.
2. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Distributions**.
3. In the **Autoinstallable Distributions** dialog, click **Create Distribution**:
 - In the **Distribution Label** field, enter a name to identify your autoinstallable distribution.
 - In the **Tree Path** field, enter the path to an installation tree located on your Uyuni Server.
 - Select the matching **Base Channel** mirrored on the Uyuni Server. This base channel must represent the distribution you want to install. It can be the **Vendor**, **Custom**, or **Cloned Channels**.
 - The **Installer Generation** should also match.
 - Optionally, you can specify kernel options which should be added when booting this distribution. There are multiple places where you can provide kernel options. Only add options here that are generic for the distribution.
4. Click [**Create Autoinstallable Distribution**].

For more information, see [**Reference > Systems >**].

Procedure: Preparing a Profile

1. In the Uyuni Web UI, navigate to **System > Autoinstallation > Profiles**.
2. In the **Autoinstallation Profiles** dialog, add the profile for your autoinstallation. It can be an **AutoYaST** or **Kickstart** profile.
3. There are two ways to create profiles:
 - Create a **Kickstart** profile using a wizard
 - Upload an externally created profile (**Kickstart** or **AutoYaST**)

For more information about the **Kickstart** wizard, see [**Reference > Systems >**].

Upload a Profile

Profiles require a label, and an **Autoinstallation Tree** (distribution).

Upload the **Kickstart** or **AutoYaST** profile. You can write your own kickstart or AutoYaST profile directly in the Web UI, or create the profile and upload it from your local file system.

AutoYaST is able to dump a profile from an existing installation. This can be used as template, but will need to be edited to make it usable by Uyuni.

Procedure: Making an AutoYaST Template Usable by Uyuni

1. Change the **add-on** section and add Uyuni URLs. The URL must use this format:

```
http://$redhat_management_server/ks/dist/child/<channel-label>/<distribution-label>
```

2. Replace **<channel-label>** and **<distribution-label>** with the correct labels. You can also use a variable for **distribution-label**. Ensure that the distribution label corresponds to the autoinstallable distribution you selected. You can only configure child channels in this file. The channels must be children of the base channel you selected in the distribution you use for this profile. You do not need to specify a base channel. The base channel is defined in the distribution.
3. Register the system after it is installed. For this step we provide script snippets that can be used.

For Salt managed clients, use the **spacewalk/minion_script** snippet:

```
<scripts>
  <init-scripts config:type="list">
    $SNIPPET('spacewalk/minion_script')
  </init-scripts>
</scripts>
```

For traditional clients, use the **spacewalk/sles_register_script** snippet:

```
<scripts>
  <init-scripts config:type="list">
    $SNIPPET('spacewalk/sles_register_script')
  </init-scripts>
</scripts>
```



For registering Salt clients, you must accept the Salt key on the Uyuni Server before you attempt autoinstallation.



If you change the distribution label, it will not automatically change the **install=** kernel option. You will need to manually change the kernel options to match the distribution label.

For more information about autoinstallation profiles, see [**Reference** > **Systems** >].

Variables and Snippets

Profiles are not finalized until they are requested by a client. This allows you to use variables in profiles. You can define profile variables in the Web UI by navigating to **Profiles** > **Variables**.

Some common variables are:

redhat_management_server

The server that all clients register to (automatically set). This applies to all clients, not just Red Hat clients.

org

The organization ID where this profile is created (automatically set).

registration_key

The key used in the registration snippets. By specifying this variable, you can set the activation key to be used to register the system.

dont_register

If specified, the registration will be skipped.

allow_config_actions

If set to **1**, it will allow traditional configuration management (traditional only).

allow_remote_commands

If set to **1**, it will allow traditional remote command execution (traditional only).

dont_disable_automatic_onlineupdate

If set, the automatic online update will stay enabled (SUSE OSes only).

dont_disable_local_repos

If set, local repositories will stay active (not recommended).

Navigate to **Systems > Autoinstallation > Autoinstallation Snippets** to see which snippets are available. For more information, see [**Reference > Systems >**].

SUSE Manager Proxy

A Uyuni Proxy is a SLE client with a special role. You can install it using AutoYaST.

1. Create an autoinstallation tree as outlined in [\[client-cfg-autoinstallation-prep\]](#).
 - Unpack a Unified Installer DVD1.
 - Use **SLE-Product-SUSE-Manager-Proxy-4.1-Pool for x86_64** as the base channel.
 - Make sure the **SUSE Manager Proxy 4.1 x86_64** product is completely mirrored.
2. Create an AutoYaST profile. You can use the following example as a starting point. Ensure you change the **<password>** to match your own settings.

You can find the AutoYaST file at <https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST>.

Then continue with [client-automating-installation.pdf](#).

Autoinstallation with AutoYaST

When you have prepared an autoinstallation profile, you can autoinstall your clients.

To start autoinstallation the client must already be known to Uyuni. You can use bare metal provisioning to bring clients into Uyuni. Alternatively, you can use the Uyuni API.

Bare Metal Provisioning

Bare metal provisioning is supported on clients with AMD or Intel x86_64 processors, and at least 1 GB of RAM.

Uyuni Server uses Cobbler over TFTP to connect to bare metal clients for provisioning. Check that you have a DHCP server and that you have set the next-server configuration parameter to match the Uyuni server IP address or hostname.

When you have the bare metal provisioning option enabled, any bare metal client connected to the Uyuni network is automatically added to the organization as soon as it is powered on. The provisioning process can take a few minutes. When it is complete, the client will be shut down, and it appears in the **Systems** list, ready to be installed.

Procedure: Provisioning Bare Metal Clients

1. In the Uyuni Web UI, navigate to **Admin > Manager Configuration > Bare-metal systems**.
2. Click **[Enable adding to this organization]**.
3. Navigate to **Systems**, locate your bare metal clients in the list, and click the client you want to provision.
4. Select the **Provisioning > Autoinstallation** tab.
5. Select the AutoYaST profile to use, and start the autoinstallation.



You cannot schedule autoinstallation for bare metal clients. Bare metal clients will be automatically installed when they are correctly configured and powered on.

New bare metal clients are added to the organization that belongs to the administrator who enabled the bare metal feature. To change the organization clients are added to, disable the bare metal feature, log in as the administrator of the new organization, and then re-enable the feature.

You can use the System Set Manager (SSM) with bare metal clients. However, not all SSM features are available for bare metal clients, because they do not yet have an operating system installed. This also applies to mixed sets that include bare metal systems. All features will become available to the set when all the clients in the set have been provisioned. For more information on SSM, see **[Client-configuration > Using-ssm >]**.

API Provisioning

You can use API calls at the command prompt to bring clients into Uyuni for autoinstallation.

Procedure: Provisioning Using the API

1. At the command prompt, use the **system.createSystemRecord** or **system.createSystemProfile** API calls. In this example, replace **<hw_addr>** with a hardware address such as **00:25:22:71:e7:c6** and **<name>** with the name of your client:

```
spacecmd api -- --args '["systemname", {"hwAddress": "<hw_addr>", "hostname":"<name>"}]'
system.createSystemProfile
```

2. In the Uyuni Web UI, navigate to **Systems**, locate your new clients in the list, and click the client you want to provision.
3. Select the **Provisioning > Autoinstallation** tab.
4. Select the AutoYaST profile to use, and start the autoinstallation. Alternatively, you can schedule the autoinstallation for a later time.

Advanced PXE Installation Configuration

If the client needs to be installed for the first time, you can use the **Create PXE installation configuration** option. This option creates a PXE boot configuration. When you power on the client, it

boots from the network and the correct profile is selected for installation.

If the client is already managed, click [**Schedule Autoinstallation and Finish**] to start the installation.

For more information about AutoYaST, see <https://doc.opensuse.org/projects/autoyast/>.

Kickstart

When you install a Red Hat Enterprise Linux client, there are a number of questions you need to answer. To automate installation, you can create a Kickstart file with all the answers to those questions, so that no user intervention is required.

Kickstart files can be kept on a server and read by individual clients during installation. The same Kickstart file is used to install multiple clients.

Kickstart can be used to schedule a registered system to be installed with a new operating system and package profile, or you can use it to install a new system that was not previously registered, or does not yet have an operating system installed.

For more information about Kickstart, see the Red Hat documentation.

Before you Begin

Some preparation is required for your infrastructure to handle Kickstart installations. Before you create a Kickstart profile, consider:

- A DHCP server is not required for kickstarting, but it can make things easier. If you are using static IP addresses, select static IP while developing your Kickstart profile.
- An FTP server can be used instead of hosting the Kickstart distribution tree using HTTP.
- If you are performing a bare metal Kickstart installation, use these settings:
 - Configure DHCP to assign the required networking parameters and the bootloader program location.
 - In the bootloader configuration file, specify the kernel and appropriate kernel options to be used.

Build a Bootable ISO

You will need to create a bootable ISO image to be used by the target system for installation. When the system is rebooted or switched on, it boots from the image, loads the Kickstart configuration from your Uyuni, and installs Red Hat Enterprise Linux according to the Kickstart profile.

Building a Bootable ISO

1. Copy the contents of **/isolinux** from the first CD-ROM of the target distribution.
2. Edit the **isolinux.cfg** file to default to 'ks'. Change the 'ks' section to read:

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

IP address-based Kickstart URLs will look like this:

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

The Kickstart distribution defined via the IP range should match the distribution from which you are building, or errors will occur.

3. OPTIONAL: If you want to use the **ksdevice**, it looks like:

```
ksdevice=eth0
```

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

4. Customize **isolinux.cfg** further as required. For example, you can add multiple options, different boot messages, or shorter timeout periods.
5. Create the ISO with this command:

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

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

6. Burn the ISO to CD-ROM and insert the disk.
7. Boot the system and type **ks** at the prompt (if you left the label for the Kickstart boot as 'ks').
8. Press *Enter* to start Kickstart.

Integrating with PXE

Instead of using a bootable ISO image, you can use a PXE image instead. This is less error-prone, allows Kickstart installation from bare metal, and integrates with existing PXE/DHCP environments.

To use this method, make sure your systems have network interface cards (NICs) that support PXE. You will need to install and configure a PXE server, ensure DHCP is running, and place the installation repository on an HTTP server that is reachable by the Uyuni Server.

Upload the Kickstart profile to the Uyuni Server using the Uyuni Web UI.

When the AutoYaST profile has been created, use the URL from the **Autoinstallation Overview** page as the image location.

For more information about PXE boot, see <https://documentation.suse.com/sles/15-SP1/html/SLES-all/cha-deployment-prep-pxe.html>.

For more information about autoinstallation profiles, see [**Reference** > **Systems** >].

Cobbler

Cobbler is an installation server that allows you to perform unattended system installations. Cobbler is installed on the Uyuni Server.



SUSE only supports Cobbler functions that are available in the Uyuni Web UI, or through the Uyuni API. Only supported features are documented here.



If you intend to use your installation with SUSE Manager for Retail formulas, do not follow this guide to configure Cobbler on the branch server. In SUSE Manager for Retail installations, the TFTPDP formula manages these settings. For more information about the TFTPDP formula, see [**Salt** > **Formula-tftpd** >].

This section explains the Cobbler features most commonly used with Uyuni:

- The **cobbler sync** command is triggered from Uyuni Server and generate the TFTP boot environment
- Installation environment analysis using the **cobbler check** 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 information about Cobbler, see <https://cobbler.readthedocs.io>.

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.



Cobbler uses host names 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 host name. Non-unique host names will cause all systems with the same host name to have the configuration files overwritten when a provisioning profile is assigned.

Configure Cobbler

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.

The PXE boot process uses DHCP to find the TFTP boot server. The Uyuni Server can act as such a TFTP boot server and Cobbler can generate the content for it. You must have administrative access to the network's DHCP server. Edit the DHCP configuration file so that it points to the Uyuni Server as the TFTP boot server:

Procedure: Example for Configuring the ISC DHCP Server

1. On the DHCP server, as root, open the **/etc/dhcpd.conf** file.
2. Append a new class with options for performing PXE boot installation. For example:

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

This example:

- Enables the **bootp** protocol for network booting.
 - Creates a class called **PXE**.
 - Identifies systems as **PXEClient** if they are configured with PXE as the first boot priority.
 - Directs PXEClients to the Cobbler server at **192.168.2.1**.
 - Retrieves the **pxelinux.0** bootloader file.
3. Save the file.

Procedure: Configuring PXE Boot in KVM

While it is possible to use KVM with PXE booting, it can be unreliable. We do not recommend you use this on production systems.

1. Use the **virsh** command to produce a dump of the current network XML description:


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

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

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

3. Use the `virsh` command to install the updated description:

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

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

Listing 1. Example: 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' />
    </dhcp>
  </ip>
</network>
```

TFTP

Uyuni uses the `tftp` daemon. The `tftp` daemon is the recommended method for PXE services, and is installed by default. The default configuration works in most cases. However, if you need to change the configuration, use the YaST Services Manager.

The TFTP service must be running so it can serve the `pxelinux.0` boot image. Start YaST and use **System > Services Manager** to configure the `tftp` daemon.

You can also synchronize Cobbler-generated TFTP contents to a Uyuni Proxy. For synchronization, HTTPS port 443 must be open.

Procedure: Installing TFTP

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

```
zypper install susemanager-tftpsync
```

2. On the Uyuni Proxy, as root user, install the `susemanager-tftpsync-recv` package:

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

Procedure: Configuring TFTP on a Proxy

1. On the Uyuni Proxy, as root, run the `configure-tftpsync.sh` script.
2. The script will interactively ask you for details on the host names and IP addresses of the Uyuni Server and Proxy, as well for the location of the `tftpboot` directory on the Proxy.

For more information, use the `configure-tftpsync.sh --help` command.

Procedure: Configuring TFTP on a Server

1. On the Uyuni Server, as root, run the `configure-tftpsync.sh` script.

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

2. Run the `cobbler sync` command to push the files to the proxy. This will fail if you have not configured the proxies correctly.
3. If you want to change the list of proxies later on, you can use the `configure-tftpsync.sh` script to edit them.



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 call `cobbler sync`.

Background Information about the Synchronization Process

A `cobbler sync` is a rebuild of every file Cobbler touched. On Uyuni, `cobbler sync` does the following actions:

1. Run pre-sync triggers. This can be any number of shell scripts.
2. Delete all files and directories that are not allowed in `/srv/www/cobbler/`.
3. Create all needed directories.
4. Delete all elements inside the directories.
5. Create the TFTP directory.

6. Write the DHCP files if management is enabled (unsupported). For more information, see [Configure Cobbler](#).
7. Do the same with DNS (unsupported).
8. Clean up the cache.
9. Run `rsync` if rsync management is enabled.
10. Run post-sync triggers. This can be any number of shell scripts (unsupported).

Uyuni also adds or removes, or edits systems that are in Cobbler. Those actions trigger a so-called lite sync process. This sync only touches files and directories that are related to the change which triggered it.



If you have created Cobbler profiles, distributions, or systems using the Uyuni Web UI, you must manage them in the Web UI. If you make changes at the command prompt, the profiles will not synchronize correctly, and the Web UI will show incorrect values.

Synchronize and Start the Cobbler Service

When `tfptpsync` is configured, the Uyuni Server must be able to access the Uyuni Proxy systems directly.



Do not start or stop the `cobblerd` service independent of the Uyuni service. Doing so can cause errors. Always use `/usr/sbin/spacewalk-service` to start or stop Uyuni.

Check that all the prerequisites the Cobbler service requires, are configured according to your requirements. You can do this by running the `cobbler check` command.

When you had to change the configuration, restart the Uyuni service:

```
/usr/sbin/spacewalk-service restart
```

Autoinstallation Templates

AutoYaST or Kickstart profiles are used to automate SUSE Linux Enterprise or Red Hat Enterprise Linux client installations. Templates are used to describe how to create autoinstallation profiles. You can create autoinstallation variables within the Uyuni Web UI. This allows you to create and manage large numbers of profiles and systems, without having to manually create profiles for each.

Cobbler uses a template engine called Cheetah that provides support for templates, variables, and snippets.

For more information on creating profiles, see [[Reference > Systems >](#)].

Kickstart Template Syntax

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 might have a networking section like 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.

Kickstart templates use syntax rules that rely on punctuation symbols. To avoid clashes, they need to be properly treated.

If the template contains shell script variables like `$(example)`, the content needs to be escaped with a backslash: `\$(example)`. If the variable is defined in the template, the templating engine will evaluate it correctly. If there is no such variable, the content will be left unchanged. Escaping the `$` 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
```

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

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

Kickstart Snippets

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

This example sets up a snippet for a common hard drive partition configuration:

```
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 vgroup=vg00 --fstype ext3 --size=5000
```

Save this snippet of the configuration to a 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')
```

Cobbler will parse the function with the snippet of code contained in the `my_partition` file.

Build ISOs with Cobbler

Cobbler can create ISO boot images that contain a set of distributions, kernels, and a menu, that work similar to a PXE installation.



Building ISOs with Cobbler is not supported on IBM Z.

The Cobbler `buildiso` command takes parameters to define the name and output location of the boot ISO. For example:

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

The boot ISO includes all profiles and systems by default. You can limit which profiles and systems are used, with the `--profiles` and `--systems` options. For example:

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



If you cannot write an ISO image to a public `tmp` directory, check your `systemd` settings in `/usr/lib/systemd/system/cobblerd.service`.

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.

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

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

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

Enable Bare Metal Systems Management

Bare metal systems management can be enabled or disabled in the Uyuni Web UI by navigating to **Admin** > **SUSE Manager Configuration** > **Bare-metal systems**.



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

When 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 when it is complete. The system will now be visible in the **Systems** > **System list**. Click on the name of the system to see basic information. For more details, go to the **Properties**, **Notes**, and **Hardware** tabs. You can migrate bare metal systems to other organizations if required, using the **Migrate** tab.

Provision 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.



System Set Manager can be used with bare metal systems. However, not all SSM features are available, because bare metal systems do not have an operating system installed. This 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.

Patch Management

You can use custom patches within your organization to manage clients. This allows you to issue patch alerts for packages in custom channels, schedule patch installation, and manage patches across organizations.

Create Patches

To use a custom patch, you need to create the patch, add packages to it and add it to one or more channels.

Procedure: Creating a Custom Patch

1. In the Uyuni Web UI, navigate to **Patches > Manage Patches**, click [**Create Patch**].
2. In the **Create Patch** section, use these details:
 - In the **Synopsis** field, type a short description of the patch.
 - In the **Advisory** field, type a label for the patch. We recommend you devise a naming convention for your organization to make patch management easier.
 - In the **Advisory Release** field, enter a release number for your patch. For example, if this is the first version of this patch, use **1**.
 - In the **Advisory Type** field, select the type of patch to use. For example, **Bug Fix Advisory** for a patch that fixes errors.
 - If you selected an advisory type of **Security Advisory**, in the **Advisory Severity** field, select the severity level to use.
 - In the **Product** field, type the name of the product this patch refers to.
 - OPTIONAL: In the **Author** field, type the name of the author of the patch.
 - Complete the **Topic**, **Description**, and **Solution** fields with further information about the patch.
3. OPTIONAL: In the **Bugs** section, specify the information of any related bugs, using these details:
 - In the **ID** field, enter the bug number.
 - In the **Summary** field, type a short description of the bug.
 - In the **Bugzilla URL** field, type the address of the bug.
 - In the **Keywords** field, type any keywords related to the bug. Use a comma between each keyword.
 - Complete the **References** and **Notes** fields with further information about the bug.
 - Select one or more channels that the new patch will be added to.
4. Click [**Create Patch**].

You can also create patches by cloning an existing one. Cloning preserves package associations and simplifies issuing patches.

Procedure: Cloning Patches

1. In the Uyuni Web UI, navigate to **Patches > Clone Patches**.
2. In the **View patches potentially applicable to:** field, select the software channel for the patch you want to clone.
3. Select the patch or patches you want to clone, and click **[Clone Patches]**.
4. Select one or more channels that the cloned patch will be added to.
5. Confirm the details to begin the clone.

When you have created a patch, you can assign packages to it.

Procedure: Assigning Packages to a Patch

1. In the Uyuni Web UI, navigate to **Patches > Manage Patches**, and click the advisory name of the patch to see the patch details.
2. Navigate to the **Packages > Add** tab.
3. In the **Channel** field, select the software channel that contains the packages you want to assign to the patch, and click **[View Packages]**. You can select **All managed packages** to see the available packages in all channels.
4. Check the packages you want to include, and click **[Add Packages]**.
5. Confirm the details of the packages, and click **[Confirm]** to assign them to the patch.
6. Navigate to the **Packages > List/Remove** tab to check that the packages have been assigned correctly.

When packages are assigned to a patch, the patch cache is updated to reflect the changes. The cache update might take a couple of minutes.

If you need to change the details of an existing patch, you can do so from the **Patches Management** page.

Procedure: Editing and Deleting Existing Patch Alerts

1. In the Uyuni Web UI, navigate to **Patches > Manage Patches**.
2. Click the advisory name of the patch to see the patch details.
3. Make the changes as required, and click **[Update Patch]**.
4. To delete a patch, select the patch in the **Patches Management** page, and click **[Delete Patches]**. Deleting patches might take a few minutes.

Apply Patches to Clients

When a patch is ready, you can apply it to clients either singly, or with other patches.

Each package within a patch is part of one or more channels. If the client is not subscribed to the channel, the update will not be installed.

If the client has a more recent version of a package already installed, the update will not be installed. If the client has an older version of the package installed, the package will be upgraded.

Procedure: Applying All Applicable Patches

1. In the Uyuni Web UI, navigate to **Systems > Overview** and select the client you want to update.
2. Navigate to the **Software > Patches** tab.
3. Click [**Select All**] to select all applicable patches.
4. Click [**Apply Patches**] to update the client.

If you are signed in with Administrator privileges, you can also perform larger batch upgrades for clients.

Procedure: Applying a Single Patch to Multiple Clients

1. In the Uyuni Web UI, navigate to **Patches > Patch List**.
2. Locate the patch you want to apply, and click the number under the **Systems** column for that patch.
3. Select the clients you want to apply the patch to, and click [**Apply Patches**]. . Confirm the list of clients to perform the update.

Procedure: Applying Multiple Patches to Multiple Clients

1. In the Uyuni Web UI, navigate to **Systems > Overview** and check the clients you want to update to add them to the system set manager.
2. Navigate to **Systems > System Set Manager** and navigate to the **Patches** tab.
3. Select the patches you want to apply to the clients and click [**Apply Patches**].
4. Schedule a date and time for the update to occur, and click [**Confirm**].
5. To check the progress of the update, navigate to **Schedule > Pending Actions**.



Scheduled package updates are installed using the contact method configured for each client. For more information, see [**Client-configuration > Contact-methods-intro >**].

System Locking

System locks are used to prevent actions from occurring on a client. For example, a system lock will prevent a client from being updated or restarted. This is useful for clients running production software, or to prevent accidental changes. You can disable the system lock when you are ready to perform actions.

System locks are implemented differently on traditional and Salt clients.

System Locks on Traditional Clients

When a traditional client is locked, no actions can be scheduled using the Web UI, and a padlock icon is displayed next to the name of the client in the **System > System List**.

Procedure: System Locking a Traditional Client

1. In the Uyuni Web UI, navigate to the **System Details** page for the client you want to lock.
2. Under **Lock Status**, click **[Lock this system]**. The client will remain locked until you click **[Unlock this system]**.

Some actions can still be completed on locked traditional clients, including remote commands, and automated patch updates. To stop automated patch updates, navigate to the **System Details** page for the client, and on the **Properties** tab, uncheck **Auto Patch Update**.

System Locks on Salt Clients

When a Salt client is locked, or put into blackout mode, no actions can be scheduled, Salt execution commands are disabled, and a yellow banner is displayed on the **System Details** page. In this mode, actions can be scheduled for the locked client using the Web UI or the API, but the actions will fail.



The locking mechanism is not available for Salt SSH clients.

Procedure: System Locking a Salt Client

1. In the Uyuni Web UI, navigate to the **System Details** page for the client you want to lock.
2. Navigate to the **Formulas** tab, check the system lock formula, and click **[Save]**.
3. Navigate to the **Formulas > System Lock** tab, check **Lock system**, and click **[Save]**. On this page, you can also enable specific Salt modules while the client is locked.
4. When you have made your changes, you might need to apply the highstate. In this case, a banner in the Web UI will notify you. The client will remain locked until you remove the system lock formula.



The system lock formula is enabled by default if SUSE CaaS Platform is detected on the node.

For more information about blackout mode in Salt, see <https://docs.saltstack.com/en/latest/topics/blackout/index.html>.

Package Locks



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

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

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

Procedure: Using Package Locks

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

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

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

Client Upgrades

Clients use the versioning system of their underlying operating system, and require regular upgrades.

For clients using SUSE operating systems, you can perform upgrades within the Uyuni Web UI. You can also automate client upgrades using the content lifecycle manager.

For more information about managing client upgrades, see [**Upgrade > Client-intro >**].

Virtualization

You can use Uyuni to manage virtualized clients in addition to regular traditional or Salt clients. In this type of installation, a virtual host is installed on the Uyuni Server to manage any number of virtual guests. If you choose to, you can install several virtual hosts to manage groups of guests.

The range of capabilities that virtualized clients have depends on the third-party virtualization provider you choose.

Xen and KVM hosts and guests can be managed directly in Uyuni. This enables you to autoinstall hosts and guests using AutoYaST or Kickstart, and manage guests in the Web UI.

For VMWare, including VMWare vSphere, Uyuni requires you to set up a virtual host manager (VHM) to control the VMs. This gives you control over the hosts and guests, but in a more limited way than available with Xen and KVM.

Other third-party virtualization providers are not directly supported by Uyuni. However, if your provider allows you to export a JSON configuration file for the VM, you can upload that configuration file to Uyuni and manage it with a VHM.

Virtualization with Xen and KVM

Xen and KVM virtualized clients can be managed directly in Uyuni.

To begin, you will need to set up a virtual host on your Uyuni Server. You can then set up autoinstallation using AutoYaST or Kickstart for future virtual hosts, and for virtual guests.

This section also includes information about administering your virtual guests after they have been installed.

Host Setup

The way that you set up Xen or KVM on a VM host depends on what operating system you want to use on its associated guests.

For SUSE operating systems, see the SLES Virtualization Guide available from <https://documentation.suse.com/sles/15-SP1/html/SLES-all/book-virt.html>.

For Red Hat Enterprise Linux operating systems, refer to the Red Hat documentation for your version.

Uyuni uses **libvirt** to install and manage guests. You must have the **libvirt** package installed on your host. In most cases, the default settings are usually sufficient, and you should not need to adjust them. However, if you want to access the VNC console on your guests as a non-root user, you will need to perform some configuration changes. For more information about how to set this up, consult the relevant documentation for your operating system.

You will require a bootstrap script on the Uyuni Server. Your bootstrap script must include the activation

key for your host. We also recommend that you include your GPG key for additional security. For more on creating a bootstrap script, see [[Client-configuration](#) > [Registration-bootstrap](#) >].

When your bootstrap script is ready, execute it on the host to register it with the Uyuni Server. For more on client registration, see [[Client-configuration](#) > [Registration-overview](#) >].

For Salt clients, you will need to enable the [Virtualization Host](#) entitlement. This allows you to see VM changes instantly. To do this, in the Uyuni Web UI, navigate to the [System Details](#) page for the host, and click on the [Properties](#) tab. Alternatively, the [Virtualization Host](#) entitlement can be added at the registration key level. In the [Add-On System Types](#) section, check [Virtualization Host](#), and click [Update Properties] to save the changes. Restart the Salt minion service to activate the change:

```
systemctl restart salt-minion
```

For traditional clients, by default, VM hosts use the [rhnsd](#) service to check for scheduled actions. The check occurs every four hours, in order to balance load in environments where there are a lot of clients. This can create delays of up to four hours before an action is carried out. When you are managing VM guests, this long delay is not always ideal, especially for actions like rebooting a guest. To address this, you can disable the [rhnsd](#) service, and enable the [osad](#) service. The [osad](#) service receives commands using a jabber protocol, and will execute commands instantly.

To disable the [rhnsd](#) service, and enable the [osad](#) daemon, run these commands as the root user:

```
service rhnsd stop
service rhnsd disable
```

```
service osad enable
service osad start
```

Autoinstallation

You can use AutoYaST or Kickstart to automatically install and register Xen and KVM guests.

You will require an activation key for the VM host you want to register the guests to, and for each guest. Your activation key must have the [provisioning](#) and [Virtualization Platform](#) entitlements. Your activation key must also have access to the [mgr-virtualization-host](#) and [mgr-osad](#) packages. For more on creating activation keys, see [[Client-configuration](#) > [Activation-keys](#) >].

If you want to automatically register the guests with Uyuni after installation, you will need to create a bootstrap script. For more on creating a bootstrap script, see [[Client-configuration](#) > [Registration-bootstrap](#) >].



Autoinstallation of VM guests works only if they are configured as Traditional clients. Salt clients can be created using a template disk image, but not by using AutoYaST or Kickstart.

Create an Autoinstallable Distribution

You will need to create an autoinstallable distribution on the VM host to be able to autoinstall clients from Uyuni. The distribution can be made available from a mounted local or remote directory, or on a loop-mounted ISO image.

The configuration of the autoinstallable distribution will differ depending on whether you are using a SLES or Red Hat Enterprise Linux operating system on your guests. The packages for a Red Hat Enterprise Linux installation are fetched from the associated base channel. Packages for installing SUSE systems are fetched from the autoinstallable distribution. Therefore, for SLES systems, the autoinstallable distribution must be a complete installation source.

Table 26. Paths for autoinstallable distributions

Operating System Type	Kernel Location	initrd Location
Red Hat Enterprise Linux	<code>images/pxeboot/vmlinuz</code>	<code>images/pxeboot/initrd.img</code>
SLES	<code>boot/<arch>/loader/initrd</code>	<code>boot/<arch>/loader/linux</code>

In all cases, ensure that the base channel matches the autoinstallable distribution.

Before you begin, ensure you have a installation media available to your VM Host. It can be on a network resource, a local directory, or an loop-mounted ISO image. Additionally, ensure that all files and directories are world-readable.

Procedure: Creating an Autoinstallable Distribution

1. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Distributions** and click [**Create Distribution**].
2. In the **Create Autoinstallable Distribution** section, use these parameters:
 - In the **Distribution Label** section, type a unique name for the distribution. Use only letters, numbers, hyphens (-), periods (.), and underscores (_), and ensure the name is longer than four characters.
 - In the **Tree Path** field, type an absolute path to the installation source.
 - In the **Base Channel** field, select the channel that matches the installation source. This channel is used as the package source for non-SUSE installations.
 - In the **Installer Generation** field, select the operating system version that matches the installation source.

- In the **Kernel Options** field, type any options to be passed to the kernel when booting for the installation. The **install=** parameter and the **self_update=0** **pt.options=self_update** parameter are added by default.
- In the **Post Kernel Options** section, type any options to be passed to the kernel when booting the installed system for the first time.

3. Click **[Create Autoinstallable Distribution]** to save.

When you have created an autoinstallable distribution, you can edit it by navigating to **Systems > Autoinstallation > Distributions** and selecting the distribution you want to edit.

Create and Upload an Autoinstallation Profile

Autoinstallation profiles contain all the installation and configuration data needed to install a system. They can also contain scripts to be executed after the installation is complete.

Kickstart profiles can be created using the Uyuni Web UI, by navigating to **Systems > Autoinstallation > Profiles**, clicking **[Create New Kickstart File]**, and following the prompts. You can also create AutoYaST or Kickstart autoinstallation profiles by hand.

An example AutoYaST profile that includes a script for registering the client with Uyuni is available in **[Client-configuration > Autoyast-example >]**. If you are using AutoYaST to install SLES, you will also need to include this snippet:

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

- For more on AutoYaST, see **[Client-configuration > Client-automating-installation >]**.
- For more on Kickstart, see **[Client-configuration > Kickstart >]**, or refer to the Red Hat documentation for your installation.

Procedure: Uploading an Autoinstallation Profile

1. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Profiles** and click **[Upload Kickstart/AutoYaST File]**.
2. In the **Create Autoinstallation Profile** section, use these parameters:
 - In the **Label** field, type a unique name for the profile. Use only letters, numbers, hyphens (-), periods (.), and underscores (_), and ensure the name is longer than six characters.
 - In the **Autoinstall Tree** field, select the autoinstallable distribution you created earlier.
 - In the **Virtualization Type** field, select the relevant Guest type (for example, **KVM Virtualized Guest**). Do not choose **Xen Virtualized Host** here.
 - **OPTIONAL:** If you want to manually create your autoinstallation profile, you can type it directly into the **File Contents** field. If you have a file already created, leave the **File**

Contents field blank.

- In the **File to Upload** field, click **[Choose File]**, and use the system dialog to select the file to upload. If the file is successfully uploaded, the filename will be shown in the **File to Upload** field.
- The contents of the uploaded file will be shown in the **File Contents** field. If you need to make edits, you can do so directly.

3. Click **[Create]** to save your changes and store the profile.

When you have created an autoinstallation profile, you can edit it by navigating to **Systems > Autoinstallation > Profiles** and selecting the profile you want to edit. Make the desired changes and save your settings by clicking **[Create]**.



If you change the **Virtualization Type** of an existing Kickstart profile, it might also modify the bootloader and partition options, potentially overwriting any custom settings. Carefully review the **Partitioning** tab to verify these settings before making changes.

Automatically Register Guests

When you install VM guests automatically, they are not registered to Uyuni. If you want your guests to be automatically registered as soon as they are installed, you can add a section to the autoinstallation profile that invokes a bootstrap script, and registers the guests.

This section gives instructions for adding a bootstrap script to an existing AutoYaST profile.

For more on creating a bootstrap script, see **[Client-configuration > Registration-bootstrap >]**. For instructions on how to do this for {kickstart}, refer to the Red Hat documentation for your installation.

Procedure: Adding a Bootstrap Script to an AutoYaST Profile

1. Ensure your bootstrap script contains the activation key for the VM guests you want to register with it, and that is located on the host at **/srv/www/htdocs/pub/bootstrap_vm_guests.sh**.
2. In the Uyuni Web UI, navigate to **Systems > Autoinstallation > Profiles**, and select the AutoYaST profile to associate this script with.
3. In the **File Contents** field, add this snippet at the end of the file, immediately before the closing **</profile>** tag. Ensure you replace the example IP address in the snippet with the correct IP address for your Uyuni Server:

```
<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>
```

4. Click **Update** to save your changes.



If your AutoYaST profile already contains a `<scripts>` section, do not add a second one. Place the bootstrap snippet inside the existing `<scripts>` section.

Autoinstall VM Guests

Once you have everything set up, you can start to autoinstall your VM guests.



Each VM host can only install one guest at a time. If you are scheduling more than one autoinstallation, make sure you time them so that the next installation does not begin before the previous one has completed. If a guest installation starts while another one is still running, the running installation will be canceled.

1. In the Uyuni Web UI, navigate to **Systems > Overview**, and select the VM host you want to install guests on.
2. Navigate to the **Virtualization** tab, and the **Provisioning** subtab.
3. Select the autoinstallation profile you want to use, and specify a unique name for the guest.
4. Choose a proxy if applicable and enter a schedule.
5. To change the guest's hardware profile and configuration options, click **[Advanced Options]**.
6. Click **[Schedule Autoinstallation and Finish]** to complete.

Manage VM Guests

You can use the Uyuni Web UI to manage your VM Guests, including actions like shutting down, restarting, and adjusting CPU and memory allocations.

To do this, you will need your Xen or KVM VM host registered to the Uyuni Server, and have the **libvirt** service running on the host. For traditional clients, you will also need the **mgr-cfg-actions** package installed on your Uyuni Server.

In the Uyuni Web UI, navigate to **Systems > System List**, and click on the VM host for the guests you want to manage. Navigate to the **Virtualization** tab to see all guests registered to this host, and access the management functions.

For more information on managing VM guests using the Web UI, see [**Reference** > **Systems** >].

Virtualization with VMWare

You can use VMWare vSphere virtual machines, including ESXi and vCenter, with Uyuni by setting up a virtual host manager (VHM).

To begin, you will need to set up a VHM on your Uyuni Server, and inventory the available VM hosts. Taskomatic can then begin data collection using the VMs API.

VHM Setup

The Virtual Host Manager (VHM) runs on the Uyuni Server.

To run a VHM, your Uyuni Server will need to have port 443 open, in order to access the VMWare API.

VMWare hosts use access roles and permissions to control access to hosts and guests. Ensure that any VMWare objects or resources that you want to be inventoried by the VHM have at least **read-only** permissions. If you want to exclude any objects or resources, mark them with **no-access**.

When you are adding new hosts to Uyuni, you will need to consider if the roles and permissions that have been assigned to users and objects need to be inventoried by Uyuni.

For more on users, roles, and permissions, see the VMWare vSphere documentation: <https://docs.vmware.com/en/VMware-vSphere/index.html>

Procedure: Creating a VMWare VHM

1. In the Uyuni Web UI, navigate to **Systems** > **Virtual Host Managers**.
2. Click [**Create**] and select **VMWare-based**.
3. In the **Add a VMWare-based Virtual Host Manager** section, use these parameters:
 - In the **Label** field, type a custom name for your VHM.
 - In the **Hostname** field, type the fully-qualified domain name (FQDN) or host IP address.
 - In the **Port** field, type the ESXi API port to use (for example, **443**).
 - In the **Username** field, type the username associated with the VM host.
 - In the **Password** field, type the password associated with the VM host user.
4. Click [**Create**] to save your changes and create the VHM.
5. On the **Virtual Host Managers** page select the new VHM.
6. On the **Properties** page, click [**Refresh Data**] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to **Systems** > **System List** > **Virtual Systems**.



Connecting to the ESXi server from a browser using HTTPS can sometimes log an **invalid certificate** error. If this occurs, refreshing the data from the virtual hosts server will fail. To correct the problem, extract the certificate from the ESXi server, and copy it to `/etc/pki/trust/anchors`. Re-trust the certificate by running the `update-ca-certificates` command on the command line, and restart the spacewalk services.

After your VHM has been created and configured, Taskomatic will run data collection automatically. If you want to manually perform data collection, navigate to **Systems > Virtual Host Managers**, select the appropriate VHM, and click **[Refresh Data]**.

Uyuni ships with a tool called `virtual-host-gatherer` that can connect to VHMs using their API, and request information about virtual hosts. `virtual-host-gatherer` maintains the concept of optional modules, where each module enables a specific VHM. This tool is automatically invoked nightly by Taskomatic. Log files for the `virtual-host-gatherer` tool are located at `/var/log/rhn/gather.log`.

Troubleshooting SSL Errors on VMWare

If you see SSL errors while configuring your VMWare installation, you need to download the CA certificate file from VMWare, and trust it on Uyuni.

Procedure: Trusting VMWare CA Certificates

1. Download the CA Certificate from your VMWare installation. You can do this by logging in to your vCenter Web UI, and clicking **[Download trusted root CA certificates]**.
2. If the downloaded CA certificates file is in `.zip` format, extract the archive. The certificate files will have a number as an extension. For example, `certificate.0`.
3. Copy the certificate files to your Uyuni Server, and save them to the `/etc/pki/trust/anchors/` directory.
4. Change the filename suffix on the copied certificate to either `.crt` or `.pem`.
5. On the Uyuni Server, at the command prompt, update the CA certificate record:

```
update-ca-certificates
```

Virtualization with Other Third Party Providers

If you want to use a third-party virtualization provider other than Xen, KVM, or VMware, you can import a JSON configuration file to Uyuni.

Similarly, if you have a VMWare installation that does not provide direct access to the API, a file-based VHM will provide you with some basic management features.



This option is for importing files that have been created with the **virtual-host-gatherer** tool. It is not designed for manually created files.

Procedure: Exporting and Importing a JSON File

1. Export the JSON configuration file by running **virtual-host-gatherer** on the VM network.
2. Save the produced file to a location accessible by your Uyuni Server.
3. In the Uyuni Web UI, navigate to **Systems > Virtual Host Managers**.
4. Click **[Create]** and select **File-based**.
5. In the **Add a file-based Virtual Host Manager** section, use these parameters:
 - In the **Label** field, type a custom name for your VHM.
 - In the **Url** field, type the path to your exported JSON configuration file.
6. Click **[Create]** to save your changes and create the VHM.
7. On the **Virtual Host Managers** page, select the new VHM.
8. On the **Properties** page, click **[Refresh Data]** to inventory the new VHM.

Listing 2. Example: Exported JSON configuration file:

```
{
  "examplevhost": {
    "10.11.12.13": {
      "cpuArch": "x86_64",
      "cpuDescription": "AMD Opteron(tm) Processor 4386",
      "cpuMhz": 3092.212727,
      "cpuVendor": "amd",
      "hostIdentifier": "'vim.HostSystem:host-182'",
      "name": "10.11.12.13",
      "os": "VMware ESXi",
      "osVersion": "5.5.0",
      "ramMb": 65512,
      "totalCpuCores": 16,
      "totalCpuSockets": 2,
      "totalCpuThreads": 16,
      "type": "vmware",
      "vms": {
        "vCenter": "564d6d90-459c-2256-8f39-3cb2bd24b7b0"
      }
    },
    "10.11.12.14": {
      "cpuArch": "x86_64",
      "cpuDescription": "AMD Opteron(tm) Processor 4386",
      "cpuMhz": 3092.212639,
      "cpuVendor": "amd",
      "hostIdentifier": "'vim.HostSystem:host-183'",
      "name": "10.11.12.14",
      "os": "VMware ESXi",
      "osVersion": "5.5.0",
      "ramMb": 65512,
      "totalCpuCores": 16,
      "totalCpuSockets": 2,
      "totalCpuThreads": 16,
      "type": "vmware",
      "vms": {
        "49737e0a-c9e6-4ceb-aef8-6a9452f67cb5": "4230c60f-3f98-2a65-f7c3-600b26b79c22",
        "5a2e4e63-a957-426b-bfa8-4169302e4fdb": "42307b15-1618-0595-01f2-427ffcdd88e",
        "NSX-gateway": "4230d43e-aafe-38ba-5a9e-3cb67c03a16a",
        "NSX-l3gateway": "4230b00f-0b21-0e9d-dfde-6c7b06909d5f",
        "NSX-service": "4230e924-b714-198b-348b-25de01482fd9"
      }
    }
  }
}
```

For more information, see the man page on your Uyuni server for **virtual-host-gatherer**:

```
man virtual-host-gatherer
```

The **README** file of that package provides background information about the **type** of a hypervisor, etc.:

```
/usr/share/doc/packages/virtual-host-gatherer/README.md
```

The man page and the **README** file also contain example configuration files.

Clusters

You can use Uyuni to directly manage SUSE CaaS Platform clusters. Manage clusters that you have registered with Uyuni by navigating to **Clusters > Overview** in the Web UI.

Autoinstallation Profile

SUSE CaaS Platform 4 provides an AutoYaST profile that you can use to autoinstall a node. The profile is in the **patterns-caasp-Management** package.

You can use Uyuni to manage one or more existing SUSE CaaS Platform clusters.



Only SUSE CaaS Platform 4 is currently supported.

Before you begin, ensure you have installed your SUSE CaaS Platform cluster.

Elect a Management Node

To manage a SUSE CaaS Platform cluster, you need to elect a client as the management node for the cluster. The management node cannot be part of the cluster, and it must have the SUSE CaaS Platform channels associated with it before you begin. You can use a single management node for multiple clusters, as long as the clusters are all of the same kind.

Procedure: Electing a Management Node

1. In the Uyuni Web UI, navigate to **Systems > System List** and click the name of the client to elect as the management node.
2. Navigate to the **Formulas > Configuration** tab, and check the **CaaSP Management Node** formula.
3. Click **[Save]** and apply the highstate.



Do not use the management node until the highstate has been completed.

List all known clusters by navigating to **Clusters > Overview**. This list displays all existing clusters, along with the cluster type, and which management node they are associated with. It also shows the nodes within the cluster, if the nodes are registered to Uyuni. For the nodes within a cluster, additional information from **skuba** and the Kubernetes API are shown, including the role, status, and whether any updates are available.

Procedure: Configuring the Management Node

1. Copy the **skuba** configuration directory from your cluster to the management node. This is the directory that the **skuba** service creates after the cluster has been bootstrapped. Take a note of the new file location for adding the cluster in the Uyuni Web UI.
2. Provide a way to authenticate.

There are two ways you can authenticate a SUSE CaaS Platform cluster, choose the method that best suits

your environment:

- Copy the passwordless private SSH key used to access the cluster nodes to the Uyuni Server, and take a note of the file location. You need the current keys and keys for any clients that you want to use in the future.
- You can use an **ssh-agent** socket, and provide the path to the socket when setting up the cluster.

There are two ways of using the **ssh-agent** with SUSE CaaS Platform. You can use **ssh-agent** locally, or forward the **ssh-agent** to the management node from another machine:

Procedure: Using a Local ssh-agent

1. On the management node, at the command prompt, start the **ssh-agent** service:

```
eval $(ssh-agent)
```

2. Add the SSH key:

```
ssh-add <key>
```

3. The socket used to access the agent is available in the **\$SSH_AUTH_SOCK** environment variable.

Procedure: Forwarding ssh-agent to the Management Node

1. On the system that is providing **ssh-agent**, at the command prompt, enable SSH forwarding, and specify the hostname of the management node:

```
ssh -A <management node>
```

2. The socket used to access the agent is available in the **\$SSH_AUTH_SOCK** environment variable.



If you are using the **ssh-agent** method, the path of the socket changes every time a new **ssh-agent** is started or a new **ssh -A** connection is started. The **ssh-agent** socket path can be updated at any time from the Uyuni Web UI. The socket path can also be overridden when starting any cluster action that requires SSH access.

Manage Clusters

To manage a cluster in Uyuni, add the cluster in the Web UI.

Procedure: Adding an Existing Cluster

1. In the Uyuni Web UI, navigate to **Clusters > Overview** and click **[Add]**.
2. Follow the prompts to provide information about your cluster, including the cluster type, and select

the management node to associate.

3. Type the path to the **skuba** configuration file for the cluster.
4. Type the passwordless SSH key you want to use, or provide the path to an **ssh-agent** socket.
5. Type a name, label, and description for the cluster.
6. Click **[Add]**.

For each cluster you manage with Uyuni, a corresponding system group is created. By default, the system group is called **Cluster <cluster_name>**. Refresh the system group to update the list of nodes. Only nodes known to Uyuni are shown.

You can remove clusters from Uyuni by navigating to **Clusters > Overview**, unchecking the cluster to be deleted, and clicking **[Delete Cluster]**.



Deleting a cluster removes the cluster from Uyuni, it does not delete the cluster nodes. Workloads running on the cluster will continue uninterrupted.

Manage Nodes

When you have the cluster created in Uyuni, you can manage nodes within the cluster.

Before you add a new node to the cluster, check the management node can access the node you want to add using passwordless SSH, or the **ssh-agent** socket you are forwarding.

You also need to ensure that the node you want to add is registered to Uyuni, and has a SUSE CaaS Platform channel assigned.

Before you add a node to a cluster, you might need to do some manual configuration. Check that the system is used exclusively as a SUSE CaaS Platform node, and that swap is turned off. For more information, see https://documentation.suse.com/suse-caasp/4.2/single-html/caasp-deployment/#_disabling_swap

Procedure: Adding Nodes to a Cluster

1. In the Uyuni Web UI, navigate to **Clusters > Overview** and click **[Join Node]**.
2. Select the nodes to add from the list of available nodes. The list of available nodes includes only nodes that are registered to Uyuni, are not management nodes, and are not currently part of any cluster.
3. Follow the prompts to enter the SUSE CaaS Platform parameters for the nodes to be added.
4. OPTIONAL: Specify a custom **ssh-agent** socket that is valid only for the nodes that are being added.
5. Click **[Save]** to schedule an action to add the nodes to the SUSE CaaS Platform cluster.

Procedure: Removing Nodes from a Cluster

1. In the Uyuni Web UI, navigate to **Clusters > Overview**, uncheck the nodes to remove, and click **[Remove Node]**.
2. Follow the prompts to define the parameters for the nodes to be removed.
3. OPTIONAL: Specify a custom **ssh-agent** socket that is valid only for the nodes that are being removed.
4. Click **[Save]** to schedule an action to remove the nodes.

For more information about node removal, see https://documentation.suse.com/suse-caasp/4/single-html/caasp-admin/#_permanent_removal.

Upgrade Clusters

If the cluster has available updates, you can use Uyuni to schedule and manage the upgrade.

Uyuni upgrades all control planes first, and then upgrades the workers. For more information, see https://documentation.suse.com/suse-caasp/4.2/single-html/caasp-admin/#_cluster_updates.

Procedure: Upgrading the Cluster

1. In the Uyuni Web UI, navigate to **Clusters > Overview**, and click the cluster to upgrade.
2. OPTIONAL: There are no SUSE CaaS Platform parameters available for you to customize for upgrade. However, you can specify a custom **ssh-agent** socket that is valid only for the nodes that are being upgraded.
3. Click **[Save]** to schedule an action to upgrade the cluster.



Uyuni will only interact with **skuba** to upgrade the cluster. Any other required action, such as configuration changes, are not issued by Uyuni.

For more information about upgrading, see https://www.suse.com/releasenotes/x86_64/SUSE-CAASP/4.

Virtual Host Managers

Virtual Host Managers (VHMs) are used to gather information from a range of client types.

VHMs can be used to collect private or public cloud instances and organize them into virtualization groups. With your virtualized clients organized this way, Taskomatic collects data on the clients for display in the Uyuni Web UI. VHMs also allow you to use subscription matching on your virtualized clients.

You can create a VHM on your Uyuni Server, and use it to inventory available public cloud instances. You can also use a VHM to manage clusters created with Kubernetes and SUSE CaaS Platform.

For more information on using a VHM with Microsoft Azure, see [**Client-configuration > Vhm-azure >**]. For more information on using a VHM with Amazon Web Services, see [**Client-configuration > Vhm-aws >**]. For more information on using a VHM with Google Compute Engine, see [**Client-configuration > Vhm-gce >**]. For more information on using a VHM with Kubernetes, see [**Client-configuration > Vhm-kubernetes >**]. For more information on using a VHM with SUSE CaaS Platform, see [**Client-configuration > Vhm-caasp >**].

SUSE Support and VM Zones

Public cloud providers use regions to define the physical geographic location of the datacenter providing virtual machines. For example, **US-East**, or **Asia**.

Regions are then further divided into zones. For example, the **US-East** region might contain zones called **us-east-2a** and **us-east-2b**, among others.

SUSE uses the zone of a virtual machine to determine the appropriate subscription to provide. If all of your VMs are provided by the same zone, you are within the terms and conditions of the **1-2 Virtual Machines** subscription.

If your VMs are provided by different zones, even if they are within the same region, you might not meet the conditions of the **1-2 Virtual Machines** subscription. In this case, check your subscription carefully.



For BYOS instances (bring your own subscription), all installed products are passed to the subscription matcher. If your public cloud instances are PAYG (pay as you go), their base products will be excluded from the subscription matcher counting.

The calculation about whether an instance is PAYG or BYOS is done at the time of registration or when a hardware refresh action is executed.

For more information, see https://www.suse.com/products/terms_and_conditions.pdf or contact SUSE.

VHM and Amazon Web Services

You can use a Uyuni VHM to gather instances from Amazon Web Services (AWS).

The VHM allows Uyuni to obtain and report information about your clusters. For more information on VHMs, see [**Client-configuration > Vhm >**].

Create an Amazon EC2 VHM

The Virtual Host Manager (VHM) runs on the Uyuni Server.

Ensure you have installed the **virtual-host-gatherer-libcloud** package on the Uyuni Server.

Procedure: Creating an Amazon EC2 VHM

1. In the Uyuni Web UI, navigate to **Systems > Virtual Host Managers**.
2. Click [**Create**] and select **Amazon EC2** from the drop-down menu.
3. In the **Add an Amazon EC2 Virtual Host Manager** section, use these parameters:
 - In the **Label** field, type a custom name for your VHM.
 - In the **Access Key ID** field, type the access key ID provided by Amazon.
 - In the **Secret Access Key** field, type the secret access key associated with the Amazon instance.
 - In the **Region** field, type the region to use.
 - In the **Zone** field, type the zone your VM is located in. This is required for subscription matching to work. For more information about setting regions and zones, see [**Client-configuration > Vhm >**].
4. Click [**Create**] to save your changes and create the VHM.
5. On the **Virtual Host Managers** page, select the new VHM.
6. On the **Properties** page, click [**Refresh Data**] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to **Systems > System List > Virtual Systems**.

Instances running on the Amazon public cloud will report a UUID to the Uyuni Server in the format of an **i** followed by seventeen hexadecimal digits:

```
i1234567890abcdef0
```

VHM and Azure

You can use a Uyuni VHM to gather instances from Microsoft Azure.

The VHM allows Uyuni to obtain and report information about your virtual machines. For more information on VHMs, see [**Client-configuration > Vhm >**].

Create an Azure VHM

The Virtual Host Manager (VHM) runs on the Uyuni Server.

Ensure you have installed the **virtual-host-gatherer-libcloud** package on the Uyuni Server.

Procedure: Creating an Azure VHM

1. In the Uyuni Web UI, navigate to **Systems > Virtual Host Managers**.
2. Click [**Create**] and select **Azure** from the drop-down menu.
3. In the **Add an Azure Virtual Host Manager** section, use these parameters:
 - In the **Label** field, type a custom name for your VHM.
 - In the **Subscription ID** field, type the subscription ID provided by Azure.
 - In the **Application ID** field, type the application ID provided by Azure.
 - In the **Tenant ID** field, type the tenant ID provided by Azure.
 - In the **Secret Key** field, type the secret key associated with the Azure instance.
 - In the **Zone** field, type the zone your VM is located in. This is required for subscription matching to work.
4. Click [**Create**] to save your changes and create the VHM.
5. On the **Virtual Host Managers** page, select the new VHM.
6. On the **Properties** page, click [**Refresh Data**] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to **Systems > System List > Virtual Systems**.

Assigning Permissions

The VHM you create needs to have the correct permissions assigned, in order for it to access the Azure VM.

Log in to your Azure account as the subscription administrator, and ensure that the Azure user account and application are in the correct groups. The group that the application is in determines the role it has, and therefore the permissions.

If the permissions are not set correctly, you might receive an error like this when you run **virtual-host-gatherer**:

```
General error: [AuthorizationFailed] The client 'client_name' with object id 'object_ID' does not have authorization to perform action 'Microsoft.Compute/virtualMachines/read' over scope '/subscriptions/not-very-secret-subscription-id' or the scope is invalid. If access was recently granted, please refresh your credentials.
```

To determine the correct credentials, run this command at the prompt on the Uyuni Server:

```
virtual-host-gatherer -i input_azure.json -o out_azure.json -vvv
```

The `input_azure.json` file should contain this information:

```
[
  {
    "id": "azure_vhm",
    "module": "Azure",
    "subscription_id": "subscription-id",
    "application_id": "application-id",
    "tenant_id": "tenant-id",
    "secret_key": "secret-key",
    "zone": "zone"
  }
]
```

Azure UUID

Instances running on the Azure public cloud will report this UUID to the Uyuni Server:

```
13f56399-bd52-4150-9748-7190aae1ff21
```

VHM and Google Compute Engine

You can use a Uyuni VHM to gather instances from Google Compute Engine (GCE).

The VHM allows Uyuni to obtain and report information about your virtual machines. For more information on VHMs, see [**Client-configuration** > **Vhm** >].

Create a GCE VHM

The Virtual Host Manager (VHM) runs on the Uyuni Server.

To run a VHM, your Uyuni Server will need to have port 443 open, in order to access the clients.

Ensure you have installed the `virtual-host-gatherer-libcloud` package on the Uyuni Server.

Before you begin, log in to the GCE panel, and download a certificate file. Store this file locally on your Uyuni Server, and take note of the path.

Procedure: Creating a GCE VHM

1. In the Uyuni Web UI, navigate to **Systems > Virtual Host Managers**.
2. Click **[Create]** and select **Google Compute Engine** from the drop-down menu.
3. In the **Add a Google Compute Engine Virtual Host Manager** section, use these parameters:
 - In the **Label** field, type a custom name for your VHM.
 - In the **Service Account Email** field, type the email address associated with your Google account.
 - In the **Cert Path** field, type the path to the certificate downloaded from the GCE panel.
 - In the **Project ID** field, type the project ID used by the GCE instance.
 - In the **Zone** field, type the zone your VM is located in. This is required for subscription matching to work.
4. Click **[Create]** to save your changes and create the VHM.
5. On the **Virtual Host Managers** page, select the new VHM.
6. On the **Properties** page, click **[Refresh Data]** to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to **Systems > System List > Virtual Systems**.

Assigning Permissions

The VHM you create needs to have the correct permissions assigned, in order for it to access the GCE VM.

Log in to your Google Cloud Platform account as an administrator, and use the Cloud Identity and Access Management (IAM) tool to ensure that the service account has the appropriate roles. You will also need to ensure that the VM has been assigned the **VM** role.

If the permissions are not set correctly, you might receive an error like this when you run **virtual-host-gatherer**:

```
ERROR: {'domain': 'global', 'reason': 'forbidden', 'message': "Required 'compute.zones.list'
permission for 'projects/project-id'"}
ERROR: Could not connect to the Google Compute Engine Public Cloud using specified
credentials.
```

To determine the correct credentials, run this command at the prompt on the Uyuni Server:

```
virtual-host-gatherer -i input_google.json -o out_google.json -vvv
```

The **input_google.json** file should contain this information:


```
[
  {
    "id": "google_vhm",
    "module": "GoogleCE",
    "service_account_email": "mail@example.com",
    "cert_path": "secret-key",
    "project_id": "project-id",
    "zone": "zone"
  }
]
```

GCE UUID

Instances running on the Google public cloud will report this UUID to Uyuni Server:

```
152986662232938449
```

VHM and Kubernetes

You can use a Uyuni VHM to manage Kubernetes clusters.

The VHM allows Uyuni to obtain and report information about your clusters. For more information on VHMs, see [**Client-configuration** > **Vhm** >].

To use Uyuni with Kubernetes, you will need to have your Uyuni Server configured for container management, with all required channels present, and a registered container build host available.

You will also require:

- At least one Kubernetes or SUSE CaaS Platform cluster available on your network.
- The **virtual-host-gatherer-Kubernetes** package installed on the Uyuni Server.
- Kubernetes version 1.5.0 or higher, or SUSE CaaS Platform.
- Docker version 1.12 or higher on the container build host.

Create a Kubernetes VHM

Kubernetes clusters are registered with Uyuni as a VHM.

You will need a **kubeconfig** file to register and authorize your Kubernetes cluster. You can get a **kubeconfig** file using the Kubernetes command line tool **kubectl**. If you are using SUSE CaaS Platform, you can download the file from the Velum interface.

Procedure: Creating a Kubernetes VHM

1. In the Uyuni Web UI, navigate to **Systems** > **Virtual Host Managers**.
2. Click [**Create**] and select **Kubernetes Cluster**.

3. In the **Add a Kubernetes Virtual Host Manager** section, use these parameters:
 - In the **Label** field, type a custom name for your VHM.
 - Select the **kubeconfig** file that contains the required data for the Kubernetes cluster.
4. In the **context** field, select the appropriate context for the cluster. This is specified in the **kubeconfig** file.
5. Click **[Create]**.

Procedure: Viewing the Nodes in a Cluster

1. In the Uyuni Web UI, navigate to **Systems > Virtual Host Managers**.
2. Select the Kubernetes cluster.
3. Refresh the node data by clicking **[Schedule refresh data]**.

The node data can take a few moments to update. You might need to refresh your browser window to see the updated information.

Any connection or authentication problems are logged to **gatherer.log**.



Node data is not refreshed during registration. You will need to manually refresh the data to see it.

Retrieve Image Runtime Data

You can view runtime data about Kubernetes images in the Uyuni Web UI, by navigating to **Images > Image List**.

The image list table contains three columns:

- **Revision:**

A sequence number that increments on every rebuild for images built by Uyuni, or on every import for externally built images.

- **Runtime:**

Overall status of the running instances for each image in registered clusters.

- **Instances:**

Number of instances running this image across all the clusters registered in Uyuni. You can see a breakdown of numbers by clicking the pop-up icon next to the number.

The **Runtime** column displays one of these status messages:

- **All instances are consistent with SUSE Manager:**

All the running instances are running the same build of the image as tracked by Uyuni.

- **Outdated instances found:**

Some of the instances are running an older build of the image. You might need to redeploy the image.

- **No information:**

The checksum of the instance image does not match the image data contained in Uyuni. You might need to redeploy the image.

Procedure: Building an Image

1. In the Uyuni Web UI, navigate to **Images > Stores**.
2. Click [**Create**] to create an image store.
3. Navigate to **Images > Profiles**.
4. Click [**Create**] to create an image profile. You will need to use a dockerfile that is suitable to deploy to Kubernetes.
5. Navigate to **Images > Build** to build an image with the new profile.
6. Deploy the image into one of the registered Kubernetes clusters. You can do this with the **kubect1** tool.

The updated data should now be available in the image list at **Images > Image List**.

Procedure: Importing a Previously Deployed Image

1. In the Uyuni Web UI, navigate to **Images > Image Stores**.
2. Add the registry that owns the image you want to import, if it is not already there.
3. Navigate to **Images > Image List** and click [**Import**].
4. Complete the fields, select the image store you created, and click [**Import**].

The imported image should now be available in the image list at **Images > Image List**.

Procedure: Rebuilding a Previously Deployed Image

1. In the Uyuni Web UI, navigate to **Images > Image List**, locate the row that contains the image you want to rebuild, and click [**Details**].
2. Navigate to the **Build Status** section, and click [**Rebuild**]. The rebuild can take some time to complete.

When the rebuild has successfully completed, the runtime status of the image is updated in the image list at **Images > Image List**. This shows that the instances are running a previous build of the image.



You can only rebuild images if they were originally built with Uyuni. You cannot rebuild imported images.

Procedure: Retrieving Additional Runtime Data

1. In the Uyuni Web UI, navigate to **Images > Image List**, locate the row that contains the running instance, and click **[Details]**.
2. Navigate to the **Overview** tab. In the **Image Info** section, there is data in the **Runtime** and **Instances** fields.
3. Navigate to the **Runtime** tab. This section contains a information about the Kubernetes pods running this image in all the registered clusters. The information in this section includes:
 - Pod name.
 - Namespace which the pod resides in.
 - The runtime status of the container in the specific pod.

Permissions and Certificates



You can only use **kubeconfig** files with Uyuni if they contain all embedded certificate data.

The API calls from Uyuni are:

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

The minimum recommended permissions for Uyuni are:

- 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"]
```

If **/pods** returns a 403 response, the entire cluster will be ignored by Uyuni.

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

VHM and SUSE CaaS Platform

You can use a Uyuni VHM to manage SUSE CaaS Platform clusters. The VHM allows Uyuni to obtain and report information about your clusters. For more information on VHMs, see [**Client-configuration > Vhm >**].



You can also manage SUSE CaaS Platform clusters directly with Uyuni, without using a VHM. For more information, see [**Client-configuration > Clusters >**].

Onboarding CaaSP Nodes

You can register each SUSE CaaS Platform node to Uyuni using the same method as you would a Salt client. For more information, see [**Client-configuration > Registration-overview >**].

We recommend that you create an activation key to associate SUSE CaaS Platform channels, and to onboard the associated nodes. For more on activation keys, see [**Client-configuration > Activation-keys >**].

If you are using **cloud-init**, we recommended that you use a bootstrap script in the **cloud-init** configuration. For more on bootstrapping, see [**Client-configuration > Registration-bootstrap >**].

When you have added the SUSE CaaS Platform nodes to Uyuni, the registered system will automatically apply the system lock Salt formula to prevent unintended actions on the client. When a system is locked, the Web UI shows a warning and you can schedule actions using the Web UI or the API, but the action will fail. For more information about system locks, see [**Client-configuration > System-locking >**].

You can disable the System Lock formula from being automatically applied by editing the configuration file. Open **/etc/rhn/rhn.conf** and add this line at the end of the file:

Add this line at the end of the **/etc/rhn/rhn.conf** file:

```
java.automatic_system_lock_cluster_nodes = false
```

Restart the spacewalk service to pick up the changes:

```
spacewalk-service restart
```

Updates related to Kubernetes are managed using the **skuba-update** tool. For more information, see <https://documentation.suse.com/suse-caasp/4/html/caasp-admin>.



When using Salt or Uyuni (either via UI or API) on any SUSE CaaS Platform nodes:

- Do not apply a patch (if the patch is marked as interactive)
- Do not mark a system to automatically install patches
- Do not perform an SP migration
- Do not reboot a node
- Do not issue any power management action via Cobbler
- Do not install a package if it breaks or conflicts the **patterns-caasp-Node-x.y**
- Do not remove a package if it breaks or conflicts or is one of the packages related with the **patterns-caasp-Node-x.y**
- Do not upgrade a package if it breaks or conflicts or is one of the packages related with the **patterns-caasp-Node-x.y**

Issuing those operations could render your SUSE CaaS Platform cluster unusable. Uyuni will not stop you from issuing these operations if the system is not locked.

Autoinstallation Profile of a SUSE CaaS Platform 4 Node

SUSE CaaS Platform 4 provides an AutoYaST profile that you can use to autoinstall a node. The profile is in the **patterns-caasp-Management** package. For more information about the profile, see https://documentation.suse.com/suse-caasp/4.2/single-html/caasp-deployment/#_autoyast_preparation.

For an example script based on the SUSE CaaS Platform 4 template, customized to make use of Uyuni, see <https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST/CaaSP-autoinstall>.

Manage a SUSE CaaS Platform Cluster With Uyuni

You can use Uyuni to manage one or more existing SUSE CaaS Platform clusters.



Only SUSE CaaS Platform 4 is currently supported.

Before you begin, ensure you have installed your SUSE CaaS Platform cluster.

Select a Management Node

To manage a SUSE CaaS Platform cluster, you need to elect a client as the management node for the cluster. The management node cannot be part of the cluster, and it must have the SUSE CaaS Platform channels associated with it before you begin. You can use a single management node for multiple clusters, as long as the clusters are all of the same kind.

Procedure: Electing a Management Node

1. In the Uyuni Web UI, navigate to **Systems > System List** and click the name of the client to elect as the management node.
2. Navigate to the **Formulas > Configuration** tab, and check the **CaaS Management Node** formula.
3. Click **[Save]** and apply the highstate.



You will not be able to use the management node until the highstate has been completed.

List all known clusters by navigating to **Clusters > Overview**. This list displays all existing clusters, along with the cluster type, and which management node they are associated with. It also shows the nodes within the cluster, if the nodes are registered to Uyuni. For the nodes within a cluster, additional information from **skuba** and the Kubernetes API are shown, including the role, status, and whether any updates are available.

For more information about the data available for nodes, see https://documentation.suse.com/suse-caasp/4/html/caasp-admin/_cluster_updates.html.

You will need to prepare the configuration from your cluster to the management node:

1. Copy the **skuba** configuration directory from your cluster to the management node. This is the directory that the **skuba** service creates after the cluster has been bootstrapped. Take a note of the new file location for adding the cluster in the Uyuni Web UI.
2. Provide a way to authentication. There are two ways you can achieve this, choose the method that best suits your environment:
 - Copy the passwordless private SSH key used to access the cluster nodes to the Uyuni Server, and take a note of the file location. You need the current keys, and keys for any clients that you want to use in the future.
 - You can use an **ssh-agent** socket, and provide the path to the socket when setting up the cluster. There are two ways of using the **ssh-agent** with SUSE CaaS Platform:
 - By using **ssh-agent** locally:
 - Start the ssh-agent locally: **eval \$(ssh-agent)**
 - Add the SSH key: **ssh-add <key>**
 - The socket used to access the agent is available in the **\$SSH_AGENT** environment variable.
 - Forward the **ssh-agent** to the management node from another machine:
 - From your source machine: **ssh -A <management node>**. The socket path is also available in the **\$SSH_AGENT** environment variable.



If you are using the **ssh-agent** method, the path of the socket changes every time a new **ssh-agent** is started or a new **ssh -A** connection is started. The **ssh-agent** socket path can be updated at any time from the Uyuni Web UI. The socket path can also be overridden when starting any cluster action that requires SSH access.

Manage Clusters

To manage a cluster in Uyuni, add the cluster in the Web UI.

Procedure: Adding an Existing Cluster

1. In the Uyuni Web UI, navigate to **Clusters > Overview** and click [**FIXME**].
2. Follow the prompts to provide information about your cluster, including the cluster type, and select the management node to associate.
3. Type the path to the **skuba** configuration file for the cluster.
4. Type the passwordless SSH key you want to use, or to the **ssh-agent** socket.
5. Type a name, label, and description for the cluster.
6. Click [**FIXME**].

For each cluster you manage with Uyuni, a corresponding system group is created. By default, the system group is called **Cluster <cluster_name>**. Refresh the system group to update the list of nodes. Only nodes known to Uyuni are shown.

You can remove clusters from Uyuni by navigating to **Clusters > Overview**, unchecking the cluster to be deleted, and clicking [**Delete Cluster**].



Deleting a cluster removes the cluster from Uyuni, it does not delete the cluster nodes. Workloads running on the cluster will continue uninterrupted.

Manage Nodes

When you have the cluster created in Uyuni, you can manage nodes within the cluster.

Before you add a new node to the cluster, check the management node can access the node you want to add using passwordless SSH, or the **ssh-agent** socket you are forwarding.

You also need to ensure that the node you want to add is registered to Uyuni, and has a SUSE CaaS Platform channel assigned.

Procedure: Adding Nodes to a Cluster

1. In the Uyuni Web UI, navigate to **Clusters > Overview** and click [**Join Node**].
2. Select the nodes to add from the list of available nodes. The list of available nodes includes only nodes that are registered to Uyuni, are not management nodes, and are not currently part of any

cluster.

3. Follow the prompts to enter the SUSE CaaS Platform parameters for the nodes to be added.
4. OPTIONAL: Specify a custom **ssh-agent** socket that is valid only for the nodes that are being added.
5. Click [**Save**] to schedule an action to add the nodes. During this action, Uyuni prepares the nodes for joining by disabling swap, then joins the nodes to the cluster.

Procedure: Removing Nodes from a Cluster

1. In the Uyuni Web UI, navigate to **Clusters > Overview**, uncheck the nodes to remove, and click [**Remove Node**].
2. Follow the prompts to define the parameters for the nodes to be removed.
3. OPTIONAL: Specify a custom **ssh-agent** socket that is valid only for the nodes that are being removed.
4. Click [**Save**] to schedule an action to remove the nodes.

For more information about node removal, see https://documentation.suse.com/suse-caasp/4/single-html/caasp-admin/#_permanent_removal.

Upgrade the Cluster

If the cluster has available updates, you can use Uyuni to schedule and manage the upgrade.

Uyuni upgrades all control planes first, and then upgrades the workers. For more information, see https://documentation.suse.com/suse-caasp/4.2/single-html/caasp-admin/#_cluster_updates.

Procedure: Upgrading the Cluster

1. In the Uyuni Web UI, navigate to **Clusters > Overview**, and click the cluster to upgrade.
2. OPTIONAL: There are no SUSE CaaS Platform parameters available for you to customize for upgrade. However, you can specify a custom **ssh-agent** socket that is valid only for the nodes that are being upgraded.
3. Click [**Save**] to schedule an action to upgrade the cluster.



Uyuni will only interact with **skuba** to upgrade the cluster. Any other required action, such as configuration changes, are not issued by Uyuni.

For more information about upgrading, see https://www.suse.com/releasenotes/x86_64/SUSE-CAASP/4.

Troubleshooting Clients

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 **mlinuz0** and **initrd0.img** files, which are provided by **pxe-default-image**, are in the locations specified in the **rhncnf** configuration file.
- Ensure the networking equipment connecting the bare metal system to the Uyuni server is working correctly, and that you can reach the Uyuni server IP address from the server.
- The bare metal system to be provisioned must have PXE booting enabled in the boot sequence, and must not be attempting to boot an operating system.
- The DHCP server must be responding to DHCP requests during boot. Check the PXE boot messages to ensure that:
 - the DHCP server is assigning the expected IP address
 - the DHCP server is assigning the the Uyuni server IP address as **next-server** for booting.
- Ensure Cobbler is running, and that the Discovery feature is enabled.

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

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



Duplicate profiles

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

Cloned Salt Clients

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

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

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

For more information and instructions, see [**Administration** > **Tshoot-registerclones** >].

Disabling the FQDNS grain

The FQDNS grain returns the list of all the fully-qualified DNS services in the system. Collecting this information is usually a fast process, but if the DNS settings have been misconfigured, it could take a much longer time. In some cases, the client could become unresponsive, or crash.

To prevent this problem, you can disable the FQDNS grain with a Salt flag. If you disable the grain, you can use a network module to provide FQDNS services, without the risk of the client becoming unresponsive.



This only applies to older Salt clients. If you registered your Salt client recently, the FQDNS grain is disabled by default.

On the Uyuni Server, at the command prompt, use this command to disable the FQDNS grain:

```
salt '*' state.sls util.mgr_disable_fqdns_grain
```

This command restarts each client and generate Salt events that the server needs to process. If you have a large number of clients, you can execute the command in batch mode instead:

```
salt --batch-size 50 '*' state.sls util.mgr_disable_fqdns_grain
```

Wait for the batch command to finish executing. Do not interrupt the process with *Ctrl+C*.

Mounting /tmp with noexec

Salt runs remote commands from **/tmp** on the client's file system. Therefore you must not mount **/tmp** with the **noexec** option.

Passing Grains to a Start Event

Every time a Salt client starts, it passes the **machine_id** grain to Uyuni. Uyuni uses this grain to determine if the client is registered. This process requires a synchronous Salt call. Synchronous Salt calls block other processes, so if you have a lot of clients start at the same time, the process could create significant delays.

To overcome this problem, a new feature has been introduced in Salt to avoid making a separate synchronous Salt call.

To use this feature, you can add a configuration parameter to the client configuration, on clients that support it.

To make this process easier, you can use the `mgr_start_event_grains.sls` helper Salt state.



This only applies to already registered clients. If you registered your Salt client recently, this config parameter is added by default.

On the Uyuni Server, at the command prompt, use this command to enable the `start_event_grains` configuration helper:

```
salt '*' state.sls util.mgr_start_event_grains
```

This command adds the required configuration into the client's configuration file, and applies it when the client is restarted. If you have a large number of clients, you can execute the command in batch mode instead:

```
salt --batch-size 50 '*' state.sls mgr_start_event_grains
```

Proxy Connections and FQDN

Sometimes clients connected through a Uyuni Proxy appear in the Web UI, but do not show that they are connected through a proxy. This can occur if you are not using the fully-qualified domain name (FQDN) to connect, and the proxy is not known to Uyuni.

To correct this behavior, specify additional FQDNs as grains in the client configuration file on the proxy:

```
grains:
  susemanager:
    custom_fqdns:
      - name.one
      - name.two
```

Registering Older Clients

To register and use CentOS 6, Oracle Linux 6, Red Hat Enterprise Linux 6, or SUSE Linux Enterprise Server with Expanded Support 6 clients, you need to configure the Uyuni Server to support older types of SSL encryption.

If you are attempting to register at the command prompt, you will see an error like this:

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

If you are attempting to register in the Web UI, you will see an error like this:

```
Rendering SLS 'base:bootstrap' failed: Jinja error: >>> No TLS 1.2 and above for RHEL6 and
SLES11. Please check your Apache config.
...
```

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

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

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

AutoYaST Example File

SUSE provides templates of AutoYaST profiles in the [SUSE/manager-build-profiles](https://github.com/SUSE/manager-build-profiles) public GitHub repository.

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:

You can find the AutoYaST file at <https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST/Minimal-AutoYaST>.

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>$c_product</product>
    <product_dir></product_dir>
  </listentry>
  <listentry>
    <!-- SLES SUSE Manager tools Pool -->
    <media_url>http://$c_server/ks/dist/child/'channel-label'/'sle-manager-
tools'/'distribution-label'</media_url>
    ...
  </listentry>
  ...
</add_on_products>
</add-on>
```

Replace **channel-label** and **distribution-label** with the correct labels (such as **sles12-sp4-updates-x86_64** and **sles12-sp4-x86_64**). Ensure that the distribution label corresponds to the Autoinstallable Distribution. Set the variables (such as **\$c_server**) according to your environment. For more information about variables, see [**Reference > Systems >**].

Here is a literal example for **sles12-sp4-x86_64**:

```
<add-on>
  <add_on_products config:type="list">
    <listentry>
      <!-- SLES12 Updates -->
      <media_url>http://192.168.150.10/ks/dist/child/dev-sles12-sp4-updates-x86_64/dev-
sles12sp4</media_url>
      <product>SLES 12 Updates</product>
      <product_dir></product_dir>
      <name>SLES12 Updates</name>
    </listentry>
    <listentry>
      <!-- SLES12 SUSE Manager Tools Pool -->
      <media_url>http://192.168.150.10/ks/dist/child/dev-sle-manager-tools12-pool-x86_64-
sp4/dev-sles12sp4</media_url>
      <product>SLES 12 Pool SUSE Manager Tools</product>
      <product_dir></product_dir>
      <name>SLES12 Pool SUSE Manager Tools</name>
    </listentry>
    <listentry>
      <!-- SLES12 SUSE Manager Tools Updates -->
      <media_url>http://192.168.150.10/ks/dist/child/dev-sle-manager-tools12-updates-x86_64-
sp4/dev-sles12sp4</media_url>
      <product>SLES 12 Updates SUSE Manager Tools</product>
      <product_dir></product_dir>
      <name>SLES12 Updates SUSE Manager Tools</name>
    </listentry>
  </add_on_products>
</add-on>
```

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 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>
```

GNU Free Documentation License

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

then there are none.

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

-
- D. Preserve all the copyright notices of the Document.
 - E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
 - F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
 - G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
 - H. Include an unaltered copy of this License.
 - I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
 - J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
 - K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
 - L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
 - M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
 - N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
 - O. Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

ADDENDUM: How to use this License for your documents

Copyright (c) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

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

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

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

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