Client Configuration Guide

Uyuni {productnumber}

Unresolved directive in nav-client-configuration-guide.pdf.en.adoc - include::./branding/pdf/entities.adoc[]

September 22 2021

Client Configuration Guide Overview

Updated: 2021-09-22

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

Uyuni is compatible with a range of client technologies: you can install traditional or Salt clients, running {sle} 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 {sle} 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:

- {check} clients running this operating system are supported by {suse}

- {cross} clients running this operating system are not supported by {suse}

- {question} clients are under consideration, and may or may not be supported at a later date.

> **NOTE**
> 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.

> **IMPORTANT**
> 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
{sle} 15	{x86_64}, {ppc64le}, {ibmz}, ARM	{check}	{check}
{sle} 12	{x86_64}, {ppc64le}, {ibmz}, ARM	{check}	{check}
{sle} 11	{x86}, {x86_64}, {ipf}, {ppc64}, {ibmz}	{check}	{check}
{sles} for SAP 15	{x86_64}, {ppc64le}	{check}	{check}
{sles} for SAP 12	{x86_64}, {ppc64le}	{check}	{check}
{micro}	{x86_64}, {ppc64le}, aarch64	{cross}	{check}
{opensuse} Leap 15	{x86_64}	{check}	{check}
{sles} ES 8	{x86_64}	{cross}	{check}

Operating System	Architecture	Traditional Clients	Salt Clients
{sles} ES 7	{x86_64}	{check}	{check}
{alibabaclo} 2	{x86_64}	{cross}	{check}
{almalinux} 8	{x86_64}	{cross}	{check}
{amazon} 2	{x86_64}	{cross}	{check}
{centos} 8	{x86_64}, {ppc64le}	{cross}	{check}
{centos} 7	{x86_64}, {ppc64le}	{check}	{check}
{rocky} 8	{x86_64}	{cross}	{check}
{debian} 10	{x86_64}, aarch64, armv7l, i586	{cross}	{check}
{debian} 9	{x86_64}, aarch64, armv7l, i586	{cross}	{check}
{oracle} 8	{x86_64}	{cross}	{check}
{oracle} 7	{x86_64}	{check}	{check}
{rhel} 8	{x86_64}	{cross}	{check}
{rhel} 7	{x86_64}	{check}	{check}
{rocky} 8	{x86_64}, {ppc64le}	{cross}	{check}
{ubuntu} 20.04	{x86_64}	{cross}	{check}
{ubuntu} 18.04	{x86_64}	{cross}	{check}

NOTE	{rhel} 6, {oracle} 6, CentOS 6, {sles}-ES 6, and {ubuntu} 16.04 are end-of-life upstream and no longer supported by {suse} as client operating systems. SUSE Manager Client Tools for these operating systems are still provided from SCC for convenience.

Supported Tools Packages

The `spacewalk-utils` and `spacewalk-utils-extras` packages can provide additional services and features.

Table 2. Spacewalk Utilities

Tool Name	Description	Supported?
spacewalk-common-channels	Add channels not provided by {scc}	{check}
spacewalk-hostname-rename	Change the hostname of the Uyuni Server	{check}
spacewalk-clone-by-date	Clone channels by a specific date	{check}
spacewalk-sync-setup	Set up ISS master and slave organization mappings	{check}
spacewalk-manage-channel-lifecycle	Manage channel lifecycles	{check}

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 {sle} operating system, including {slsa}, {sleda}, {sles} for SAP, and {sles} for HPC.

IMPORTANT	The operating system you run on a client is supported by the organization that supplies the operating system. {sle} is supported by {suse}. {opensuse} is supported by the {suse} community.

The icons in this table indicate:

- {check} the feature is available on both Salt and traditional clients
- {cross} the feature is not available
- {question} the feature is under consideration, and may or may not be made available at a later date
- {trad} 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	{sle} 11	{sle} 12	{sle} 15	{opensuse} 15
Client	{check}	{check}	{check}	{check}
System packages	{suse}	{suse}	{suse}	{opensuse} Community
Registration	{check}	{check}	{check}	{salt}
Install packages	{check}	{check}	{check}	{salt}
Apply patches	{check}	{check}	{check}	{salt}

Feature	{sle} 11	{sle} 12	{sle} 15	{opensuse} 15
Remote commands	{check}	{check}	{check}	{salt}
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)	{check}	{check}	{check}	{salt}
Product migration	{check}	{check}	{check}	{salt}
Basic Virtual Guest Management {star}	{trad}	{check}	{check}	{salt}
Advanced Virtual Guest Management {star}	{cross}	{salt}	{salt}	{salt}
Virtual Guest Installation ({ay}), as Host OS	{trad}	{trad}	{trad}	{cross}
Virtual Guest Installation (image template), as Host OS	{cross}	{salt}	{salt}	{salt}
Virtual Guest Management	{cross}	{salt}	{salt}	{salt}
System deployment (PXE/{ay})	{check}	{check}	{check}	{check}
System redeployment ({ay})	{trad}	{check}	{check}	{salt}
Contact methods	{trad}: OSAD, RHNSD, SSH-push. {salt}: ZeroMQ, Salt-SSH	{trad}: OSAD, RHNSD, SSH-push. {salt}: ZeroMQ, Salt-SSH	{trad}: OSAD, RHNSD, SSH-push. {salt}: ZeroMQ, Salt-SSH	{salt}: ZeroMQ, Salt-SSH

Feature	{sle} 11	{sle} 12	{sle} 15	{opensuse} 15
Works with Uyuni Proxy	{check}	{check}	{check}	{salt}
Action chains	{check}	{check}	{check}	{salt}
Staging (pre-download of packages)	{check}	{check}	{check}	{salt}
Duplicate package reporting	{check}	{check}	{check}	{salt}
CVE auditing	{check}	{check}	{check}	{salt}
SCAP auditing	{check}	{check}	{check}	{salt}
Package verification	{trad}	{trad}	{trad}	{cross}
Package locking	{salt}	{salt}	{salt}	{salt}
System locking	{trad}	{trad}	{trad}	{cross}
Maintenance Windows	{check}	{check}	{check}	{check}
System snapshot	{trad}	{trad}	{trad}	{cross}
Configuration file management	{check}	{check}	{check}	{salt}
Package profiles	{trad}. {salt}: Profiles supported, Sync not supported	{trad}. {salt}: Profiles supported, Sync not supported	{trad}. {salt}: Profiles supported, Sync not supported	{salt}: Profiles supported, Sync not supported
Power management	{check}	{check}	{check}	{check}
Monitoring	{question}	{salt}	{salt}	{salt}
Docker buildhost	{cross}	{salt}	{salt}	{question}
Build Docker image with OS	{cross}	{salt}	{salt}	{salt}
Kiwi buildhost	{cross}	{salt}	{question}	{question}
Build Kiwi image with OS	{cross}	{salt}	{question}	{cross}
Recurring Actions	{salt}	{salt}	{salt}	{salt}
AppStreams	N/A	N/A	N/A	N/A

Feature	{sle} 11	{sle} 12	{sle} 15	{opensuse} 15
Yomi	{check}	{check}	{check}	{check}

{star} 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 all features of Basic virtual guest management plus fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported {sleses} Features

This table lists the availability of various features on {sleses} clients.

> **IMPORTANT**
> The operating system you run on a client is supported by the organization that supplies the operating system. {sleses} is supported by {suse}.

The icons in this table indicate:

- {check} the feature is available on both Salt and traditional clients
- {cross} the feature is not available
- {question} the feature is under consideration, and may or may not be made available at a later date
- {trad} the feature is supported only on traditional clients
- {salt} the feature is supported only on Salt clients.

Table 4. Supported Features on {sleses} Operating Systems

Feature	{slsa} ES 7	{slsa} ES 8
Client	{check}	{salt}
System packages	{suse}	{suse}
Registration	{check}	{salt}
Install packages	{check}	{salt}
Apply patches	{check}	{salt}
Remote commands	{check}	{salt}
System package states	{salt}	{salt}

Feature	{slsa} ES 7	{slsa} ES 8
System custom states	{salt}	{salt}
Group custom states	{salt}	{salt}
Organization custom states	{salt}	{salt}
System set manager (SSM)	{salt}	{salt}
Product migration	N/A	N/A
Basic Virtual Guest Management {star}	{check}	{salt}
Advanced Virtual Guest Management {star}	{salt}	{salt}
Virtual Guest Installation (Kickstart), as Host OS	{trad}	{cross}
Virtual Guest Installation (image template), as Host OS	{check}	{salt}
System deployment (PXE/Kickstart)	{check}	{salt}
System redeployment (Kickstart)	{check}	{cross}
Contact methods	{trad}: OSAD, RHNSD, SSH-push. {salt}: ZeroMQ, Salt-SSH	{salt}: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	{check}	{salt}
Action chains	{check}	{salt}
Staging (pre-download of packages)	{check}	{salt}
Duplicate package reporting	{check}	{salt}
CVE auditing	{check}	{salt}
SCAP auditing	{check}	{salt}
Package verification	{trad}	{cross}
Package locking	{check}	{question}
System locking	{trad}	{cross}
Maintenance Windows	{check}	{check}
System snapshot	{trad}	{salt}

Feature	{slsa} ES 7	{slsa} ES 8
Configuration file management	{check}	{salt}
Snapshots and profiles	{trad}. {salt}: Profiles supported, Sync not supported	{salt}: Profiles supported, Sync not supported
Power management	{check}	{salt}
Monitoring	{salt}	{salt}
Docker buildhost	{cross}	{cross}
Build Docker image with OS	{question}	{question}
Kiwi buildhost	{cross}	{cross}
Build Kiwi image with OS	{cross}	{cross}
Recurring Actions	{salt}	{salt}
AppStreams	N/A	{check}
Yomi	N/A	N/A

{star} 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 all features of Basic virtual guest management plus fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported {micro} and {opensuse} {microos} Client Features

WARNING

Support for {micro} and {opensuse} {microos} clients is provided as a technology preview for testing purposes, and not all features are fully functional at this stage. This feature is expected to be fully supported in a later version of Uyuni. Do not use this feature on production systems.

IMPORTANT

The operating system you run on a client is supported by the organization that supplies the operating system. {micro} is supported by {suse}. {opensuse} {microos} is supported by the {suse} community.

The icons in this table indicate:

- {check} the feature is available on both Salt and traditional clients

- {cross} the feature is not available

- {question} the feature is under consideration, and may or may not be made available at a later date

- {trad} the feature is supported only on traditional clients

- {salt} the feature is supported only on Salt clients.

Table 5. Supported Features on {micro} and {opensuse} {microos} Operating Systems

Feature	{micro} and {opensuse} {microos}
Client	{salt}
Operating system packages	{salt}
Registration	{salt}
Install packages	{salt}
Apply patches (requires CVE ID)	{salt}
Remote commands	{salt}
System package states	{salt}
System custom states	{salt}
Group custom states	{salt}
Organization custom states	{salt}
System set manager (SSM)	{salt}
Product migration	{question}
Basic Virtual Guest Management {star}	{question}
Advanced Virtual Guest Management {star}	{question}
Virtual Guest Installation (Kickstart), as Host OS	{cross}
Virtual Guest Installation (image template), as Host OS	{question}
System deployment (PXE/Kickstart)	{question}
System redeployment (Kickstart)	{cross}
Contact methods	{salt}: ZeroMQ
Works with Uyuni Proxy	{salt}
Action chains	{question}

Feature	{micro} and {opensuse} {microos}
Staging (pre-download of packages)	{question}
Duplicate package reporting	{salt}
CVE auditing (requires CVE ID)	{salt}
SCAP auditing	{question}
Package verification	{question}
Package locking	{salt}
System locking	{question}
Maintenance Windows	{question}
System snapshot	{cross}
Configuration file management	{salt}
Snapshots and profiles	{salt}: Profiles supported, Sync not supported
Power management	{salt}
Monitoring	{salt}
Docker buildhost	{cross}
Build Docker image with OS	{cross}
Kiwi buildhost	{cross}
Build Kiwi image with OS	{salt}
Recurring Actions	{salt}
AppStreams	N/A
Yomi	{question}

{star} 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 all features of Basic virtual guest management plus fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported {alibabaclo} Features

This table lists the availability of various features on {alibabaclo} clients.

> **IMPORTANT** The operating system you run on a client is supported by the organization that supplies the operating system. {alibabaclo} is supported by Alibaba Cloud.

The icons in this table indicate:

- {check} the feature is available on both Salt and traditional clients

- {cross} the feature is not available

- {question} the feature is under consideration, and may or may not be made available at a later date

- {trad} the feature is supported only on traditional clients

- {salt} the feature is supported only on Salt clients

Table 6. Supported Features on {alibabaclo} Operating Systems

Feature	{alibabaclo} 2
Client	{salt}
Operating system packages	{salt}
Registration	{salt}
Install packages	{salt}
Apply patches (requires CVE ID)	{salt}
Remote commands	{salt}
System package states	{salt}
System custom states	{salt}
Group custom states	{salt}
Organization custom states	{salt}
System set manager (SSM)	{salt}
Product migration	N/A
Basic Virtual Guest Management {star}	{question}
Advanced Virtual Guest Management {star}	{question}
Virtual Guest Installation (Kickstart), as Host OS	{cross}

Feature	{alibabaclo} 2
Virtual Guest Installation (image template), as Host OS	{question}
System deployment (PXE/Kickstart)	{question}
System redeployment (Kickstart)	{question}
Contact methods	{salt}: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	{salt}
Action chains	{salt}
Staging (pre-download of packages)	{salt}
Duplicate package reporting	{salt}
CVE auditing (requires CVE ID)	{salt}
SCAP auditing	{salt}
Package verification	{cross}
Package locking	{cross}
System locking	{cross}
Maintenance Windows	{check}
System snapshot	{cross}
Configuration file management	{salt}
Snapshots and profiles	{salt}: Profiles supported, Sync not supported
Power management	{question}
Monitoring	{salt}
Docker buildhost	{salt}
Build Docker image with OS	{salt}
Kiwi buildhost	{salt}
Build Kiwi image with OS	{salt}
Recurring Actions	{salt}
AppStreams	N/A
Yomi	N/A

{star} 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 all features of Basic virtual guest management plus fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

{star} The traditional stack is available on {alibabaclo} but it is unsupported.

Supported {almalinux} Features

This table lists the availability of various features on {almalinux} clients.

IMPORTANT	The operating system you run on a client is supported by the organization that supplies the operating system. {almalinux} is supported by the {almalinux} community.

The icons in this table indicate:

- {check} the feature is available on both Salt and traditional clients
- {cross} the feature is not available
- {question} the feature is under consideration, and may or may not be made available at a later date
- {trad} the feature is supported only on traditional clients
- {salt} the feature is supported only on Salt clients.

Table 7. Supported Features on {almalinux} Operating Systems

Feature	{almalinux} 8
Client	{salt} (plain {almalinux})
System packages	{almalinux} Community
Registration	{salt}
Install packages	{salt}
Apply patches	{salt}
Remote commands	{salt}
System package states	{salt}
System custom states	{salt}

Feature	{almalinux} 8
Group custom states	{salt}
Organization custom states	{salt}
System set manager (SSM)	{salt}
Product migration	N/A
Basic Virtual Guest Management {star}	{salt}
Advanced Virtual Guest Management {star}	{salt}
Virtual Guest Installation (Kickstart), as Host OS	{cross}
Virtual Guest Installation (image template), as Host OS	{salt}
System deployment (PXE/Kickstart)	{salt}
System redeployment (Kickstart)	{salt}
Contact methods	{salt}: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	{salt}
Action chains	{salt}
Staging (pre-download of packages)	{salt}
Duplicate package reporting	{salt}
CVE auditing	{salt}
SCAP auditing	{salt}
Package verification	{cross}
Package locking	{cross}
System locking	{cross}
Maintenance Windows	{check}
System snapshot	{cross}
Configuration file management	{salt}
Snapshots and profiles	{salt}: Profiles supported, Sync not supported
Power management	{salt}
Monitoring	{salt}

Feature	{almalinux} 8
Docker buildhost	{cross}
Build Docker image with OS	{cross}
Kiwi buildhost	{cross}
Build Kiwi image with OS	{cross}
Recurring Actions	{salt}
AppStreams	{check}
Yomi	N/A

{star} 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 all features of Basic virtual guest management plus fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported {amazon} Features

This table lists the availability of various features on {amazon} clients.

> **IMPORTANT** The operating system you run on a client is supported by the organization that supplies the operating system. {amazon} is supported by Amazon.

The icons in this table indicate:

- {check} the feature is available on both Salt and traditional clients

- {cross} the feature is not available

- {question} the feature is under consideration, and may or may not be made available at a later date

- {trad} the feature is supported only on traditional clients

- {salt} the feature is supported only on Salt clients

Table 8. Supported Features on {amazon} Operating Systems

Feature	{amazon} 2
Client	{salt}

Feature	{amazon} 2
Operating system packages	{salt}
Registration	{salt}
Install packages	{salt}
Apply patches (requires CVE ID)	{salt}
Remote commands	{salt}
System package states	{salt}
System custom states	{salt}
Group custom states	{salt}
Organization custom states	{salt}
System set manager (SSM)	{salt}
Product migration	N/A
Basic Virtual Guest Management {star}	{question}
Advanced Virtual Guest Management {star}	{question}
Virtual Guest Installation (Kickstart), as Host OS	{cross}
Virtual Guest Installation (image template), as Host OS	{question}
System deployment (PXE/Kickstart)	{question}
System redeployment (Kickstart)	{question}
Contact methods	{salt}: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	{salt}
Action chains	{salt}
Staging (pre-download of packages)	{salt}
Duplicate package reporting	{salt}
CVE auditing (requires CVE ID)	{salt}
SCAP auditing	{salt}
Package verification	{cross}
Package locking	{cross}

Feature	{amazon} 2
System locking	{cross}
Maintenance Windows	{check}
System snapshot	{cross}
Configuration file management	{salt}
Snapshots and profiles	{salt}: Profiles supported, Sync not supported
Power management	{question}
Monitoring	{salt}
Docker buildhost	{salt}
Build Docker image with OS	{salt}
Kiwi buildhost	{salt}
Build Kiwi image with OS	{salt}
Recurring Actions	{salt}
AppStreams	N/A
Yomi	N/A

{star} 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 all features of Basic virtual guest management plus fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

{star} The traditional stack is available on {amazon} but it is unsupported.

Supported CentOS Features

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

IMPORTANT	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:

- {check} the feature is available on both Salt and traditional clients

- {cross} the feature is not available

- {question} the feature is under consideration, and may or may not be made available at a later date

- {trad} the feature is supported only on traditional clients

- {salt} the feature is supported only on Salt clients.

Table 9. Supported Features on {centos} Operating Systems

Feature	{centos} 7	{centos} 8
Client	{check} (plain CentOS)	{salt} (plain CentOS)
System packages	{centos} Community	{centos} Community
Registration	{check}	{salt}
Install packages	{check}	{salt}
Apply patches (requires CVE ID)	{check} (third-party service required for errata)	{salt} (third-party service required for errata)
Remote commands	{check}	{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)	{check}	{salt}
Product migration	N/A	N/A
Basic Virtual Guest Management {star}	{check}	{salt}
Advanced Virtual Guest Management {star}	{salt}	{salt}
Virtual Guest Installation (Kickstart), as Host OS	{trad}	{cross}
Virtual Guest Installation (image template), as Host OS	{check}	{salt}
System deployment (PXE/Kickstart)	{check}	{salt}

Feature	{centos} 7	{centos} 8
System redeployment (Kickstart)	{check}	{salt}
Contact methods	{trad}: OSAD, RHNSD, SSH-push. {salt}: ZeroMQ, Salt-SSH	{salt}: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	{check}	{salt}
Action chains	{check}	{salt}
Staging (pre-download of packages)	{check}	{salt}
Duplicate package reporting	{check}	{salt}
CVE auditing (requires CVE ID)	{check}	{salt}
SCAP auditing	{check}	{salt}
Package verification	{trad}	{cross}
Package locking	{check}	{question}
System locking	{trad}	{cross}
Maintenance Windows	{check}	{check}
System snapshot	{trad}	{cross}
Configuration file management	{check}	{salt}
Snapshots and profiles	{trad}. {salt}: Profiles supported, Sync not supported	{salt}: Profiles supported, Sync not supported
Power management	{check}	{salt}
Monitoring	{salt}	{salt}
Docker buildhost	{cross}	{cross}
Build Docker image with OS	{cross}	{cross}
Kiwi buildhost	{cross}	{cross}
Build Kiwi image with OS	{cross}	{cross}
Recurring Actions	{salt}	{salt}
AppStreams	N/A	{check}
Yomi	N/A	N/A

{star} 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 all features of Basic virtual guest management plus 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.

IMPORTANT	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:

- {check} the feature is available on both Salt and traditional clients
- {cross} the feature is not available
- {question} the feature is under consideration, and may or may not be made available at a later date
- {trad} the feature is supported only on traditional clients
- {salt} the feature is supported only on Salt clients.

Table 10. Supported Features on {debian} Operating Systems

Feature	**{debian} 9**	**{debian} 10**
Client	{check}	{check}
System packages	{debian} Community	{debian} Community
Registration	{salt}	{salt}
Install packages	{salt}	{salt}
Apply patches	{question}	{question}
Remote commands	{salt}	{salt}
System package states	{salt}	{salt}
System custom states	{salt}	{salt}
Group custom states	{salt}	{salt}
Organization custom states	{salt}	{salt}

Feature	{debian} 9	{debian} 10
System set manager (SSM)	{salt}	{salt}
Product migration	N/A	N/A
Basic Virtual Guest Management {star}	{salt}	{salt}
Advanced Virtual Guest Management {star}	{salt}	{salt}
Virtual Guest Installation (Kickstart), as Host OS	{cross}	{cross}
Virtual Guest Installation (image template), as Host OS	{salt}	{salt}
System deployment (PXE/Kickstart)	{cross}	{cross}
System redeployment (Kickstart)	{cross}	{cross}
Contact methods	{salt}: ZeroMQ, Salt-SSH	{salt}: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	{salt}	{salt}
Action chains	{salt}	{salt}
Staging (pre-download of packages)	{salt}	{salt}
Duplicate package reporting	{salt}	{salt}
CVE auditing	{question}	{question}
SCAP auditing	{question}	{question}
Package verification	{cross}	{cross}
Package locking	{cross}	{cross}
System locking	{cross}	{cross}
Maintenance Windows	{check}	{check}
System snapshot	{cross}	{cross}
Configuration file management	{salt}	{salt}
Package profiles	{salt}: Profiles supported, Sync not supported	{salt}: Profiles supported, Sync not supported
Power management	{check}	{check}

Feature	{debian} 9	{debian} 10
Monitoring	{salt}	{salt}
Docker buildhost	{question}	{question}
Build Docker image with OS	{salt}	{salt}
Kiwi buildhost	{cross}	{cross}
Build Kiwi image with OS	{cross}	{cross}
Recurring Actions	{salt}	{salt}
AppStreams	N/A	N/A
Yomi	N/A	N/A

{star} 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 all features of Basic virtual guest management plus 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} clients.

> **IMPORTANT** The operating system you run on a client is supported by the organization that supplies the operating system. {oracle} is supported by Oracle.

The icons in this table indicate:

- {check} the feature is available on both Salt and traditional clients
- {cross} the feature is not available
- {question} the feature is under consideration, and may or may not be made available at a later date
- {trad} the feature is supported only on traditional clients
- {salt} the feature is supported only on Salt clients

Table 11. Supported Features on {oracle} Operating Systems

Feature	{oracle} 6	{oracle} 7	{oracle} 8
Client	{check}	{check}	{salt}
Operating system packages	{check}	{check}	{salt}
Registration	{check}	{check}	{salt}
Install packages	{check}	{check}	{salt}
Apply patches (requires CVE ID)	{check}	{check}	{salt}
Remote commands	{check}	{check}	{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)	{check}	{check}	{salt}
Product migration	N/A	N/A	N/A
Basic Virtual Guest Management {star}	{trad}	{check}	{salt}
Advanced Virtual Guest Management {star}	{cross}	{salt}	{salt}
Virtual Guest Installation (Kickstart), as Host OS	{trad}	{trad}	{cross}
Virtual Guest Installation (image template), as Host OS	{trad}	{check}	{salt}
System deployment (PXE/Kickstart)	{check}	{check}	{salt}
System redeployment (Kickstart)	{trad}	{check}	{salt}
Contact methods	{trad}: OSAD, RHNSD, SSH-push. {salt}: ZeroMQ, Salt-SSH	{trad}: OSAD, RHNSD, SSH-push. {salt}: ZeroMQ, Salt-SSH	{salt}: ZeroMQ, Salt-SSH

Feature	{oracle} 6	{oracle} 7	{oracle} 8
Works with Uyuni Proxy	{check}	{check}	{salt}
Action chains	{check}	{check}	{salt}
Staging (pre-download of packages)	{check}	{check}	{salt}
Duplicate package reporting	{check}	{check}	{salt}
CVE auditing (requires CVE ID)	{check}	{check}	{salt}
SCAP auditing	{check}	{check}	{salt}
Package verification	{trad}	{trad}	{cross}
Package locking	{trad}	{check}	{question}
System locking	{trad}	{trad}	{cross}
Maintenance Windows	{check}	{check}	{check}
System snapshot	{trad}	{trad}	{cross}
Configuration file management	{check}	{check}	{salt}
Snapshots and profiles	{trad}. {salt}: Profiles supported, Sync not supported	{trad}. {salt}: Profiles supported, Sync not supported	{salt}: Profiles supported, Sync not supported
Power management	{check}	{check}	{salt}
Monitoring	{salt}	{salt}	{salt}
Docker buildhost	{cross}	{cross}	{cross}
Build Docker image with OS	{cross}	{cross}	{cross}
Kiwi buildhost	{cross}	{cross}	{cross}
Build Kiwi image with OS	{cross}	{cross}	{cross}
Recurring Actions	{salt}	{salt}	{salt}
AppStreams	N/A	N/A	{check}
Yomi	N/A	N/A	N/A

{star} 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 all features of Basic virtual guest management plus fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported {rhel} Features

This table lists the availability of various features on native {rhel} clients (without Expanded Support).

IMPORTANT	The operating system you run on a client is supported by the organization that supplies the operating system. {rhel} is supported by {redhat}.

The icons in this table indicate:

- {check} the feature is available on both Salt and traditional clients
- {cross} the feature is not available
- {question} the feature is under consideration, and may or may not be made available at a later date
- {trad} the feature is supported only on traditional clients
- {salt} the feature is supported only on Salt clients.

Table 12. Supported Features on {rhel} Operating Systems

Feature	{rhela} 6	{rhela} 7	{rhela} 8
Client	{check}	{check}	{salt}
System packages	{redhat}	{redhat}	{redhat}
Registration	{check}	{check}	{salt}
Install packages	{check}	{check}	{salt}
Apply patches	{check}	{check}	{salt}
Remote commands	{check}	{check}	{salt}
System package states	{salt}	{salt}	{salt}
System custom states	{salt}	{salt}	{salt}
Group custom states	{salt}	{salt}	{salt}

Feature	{rhela} 6	{rhela} 7	{rhela} 8
Organization custom states	{salt}	{salt}	{salt}
System set manager (SSM)	{salt}	{salt}	{salt}
Product migration	N/A	N/A	N/A
Basic Virtual Guest Management {star}	{trad}	{check}	{salt}
Advanced Virtual Guest Management {star}	{cross}	{salt}	{salt}
Virtual Guest Installation (Kickstart), as Host OS	{trad}	{trad}	{cross}
Virtual Guest Installation (image template), as Host OS	{trad}	{check}	{salt}
System deployment (PXE/Kickstart)	{check}	{check}	{salt}
System redeployment (Kickstart)	{trad}	{check}	{salt}
Contact methods	{trad}: OSAD, RHNSD, SSH-push. {salt}: ZeroMQ, Salt-SSH	{trad}: OSAD, RHNSD, SSH-push. {salt}: ZeroMQ, Salt-SSH	{salt}: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	{check}	{check}	{salt}
Action chains	{check}	{check}	{salt}
Staging (pre-download of packages)	{check}	{check}	{salt}
Duplicate package reporting	{check}	{check}	{salt}
CVE auditing	{check}	{check}	{salt}
SCAP auditing	{check}	{check}	{salt}
Package verification	{trad}	{trad}	{cross}
Package locking	{trad}	{check}	{question}
System locking	{trad}	{trad}	{cross}

Feature	{rhela} 6	{rhela} 7	{rhela} 8
Maintenance Windows	{check}	{check}	{check}
System snapshot	{trad}	{trad}	{cross}
Configuration file management	{check}	{check}	{salt}
Snapshots and profiles	{trad}. {salt}: Profiles supported, Sync not supported	{trad}. {salt}: Profiles supported, Sync not supported	{salt}: Profiles supported, Sync not supported
Power management	{check}	{check}	{salt}
Monitoring	{salt}	{salt}	{salt}
Docker buildhost	{cross}	{cross}	{cross}
Build Docker image with OS	{question}	{question}	{question}
Kiwi buildhost	{cross}	{cross}	{cross}
Build Kiwi image with OS	{cross}	{cross}	{cross}
Recurring Actions	{salt}	{salt}	{salt}
AppStreams	N/A	N/A	{check}
Yomi	N/A	N/A	N/A

{star} 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 all features of Basic virtual guest management plus fast refresh, VM lifecycle actions (delete, reset, power off), modifying VM disk, network, graphical display, and graphical display configuration.

Supported {rocky} Features

This table lists the availability of various features on {rocky} clients.

	The operating system you run on a client is supported by the organization that supplies the operating system. {rocky} is supported by the {rocky} community.
IMPORTANT	

The icons in this table indicate:

- {check} the feature is available on both Salt and traditional clients
- {cross} the feature is not available
- {question} the feature is under consideration, and may or may not be made available at a later date
- {trad} the feature is supported only on traditional clients
- {salt} the feature is supported only on Salt clients.

Table 13. Supported Features on {rocky} Operating Systems

Feature	{rocky} 8
Client	{salt} (plain {rocky})
System packages	{rocky} Community
Registration	{salt}
Install packages	{salt}
Apply patches	{salt}
Remote commands	{salt}
System package states	{salt}
System custom states	{salt}
Group custom states	{salt}
Organization custom states	{salt}
System set manager (SSM)	{salt}
Product migration	N/A
Basic Virtual Guest Management {star}	{salt}
Advanced Virtual Guest Management {star}	{salt}
Virtual Guest Installation (Kickstart), as Host OS	{cross}
Virtual Guest Installation (image template), as Host OS	{salt}
System deployment (PXE/Kickstart)	{salt}

Feature	{rocky} 8
System redeployment (Kickstart)	{salt}
Contact methods	{salt}: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	{salt}
Action chains	{salt}
Staging (pre-download of packages)	{salt}
Duplicate package reporting	{salt}
CVE auditing	{salt}
SCAP auditing	{salt}
Package verification	{cross}
Package locking	{question}
System locking	{cross}
Maintenance Windows	{check}
System snapshot	{cross}
Configuration file management	{salt}
Snapshots and profiles	{salt}: Profiles supported, Sync not supported
Power management	{salt}
Monitoring	{salt}
Docker buildhost	{cross}
Build Docker image with OS	{cross}
Kiwi buildhost	{cross}
Build Kiwi image with OS	{cross}
Recurring Actions	{salt}
AppStreams	{check}
Yomi	N/A

{star} 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 all features of Basic virtual guest management plus 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.

> **IMPORTANT** 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:

- {check} the feature is available on both Salt and traditional clients
- {cross} the feature is not available
- {question} the feature is under consideration, and may or may not be made available at a later date
- {trad} the feature is supported only on traditional clients
- {salt} the feature is supported only on Salt clients.

Table 14. Supported Features on {ubuntu} Operating Systems

Feature	{ubuntu} 16.04	{ubuntu} 18.04	{ubuntu} 20.04
Client	{check}	{check}	{check}
System packages	Canonical	Canonical	Canonical
Registration	{salt}	{salt}	{salt}
Install packages	{salt}	{salt}	{salt}
Apply patches	{question}	{question}	{question}
Remote commands	{salt}	{salt}	{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}

Feature	{ubuntu} 16.04	{ubuntu} 18.04	{ubuntu} 20.04
Product migration	N/A	N/A	N/A
Basic Virtual Guest Management {star}	{salt}	{salt}	{salt}
Advanced Virtual Guest Management {star}	{salt}	{salt}	{salt}
Virtual Guest Installation (Kickstart), as Host OS	{cross}	{cross}	{cross}
Virtual Guest Installation (image template), as Host OS	{salt}	{salt}	{salt}
System deployment (PXE/Kickstart)	{cross}	{cross}	{cross}
System redeployment (Kickstart)	{cross}	{cross}	{cross}
Contact methods	{salt}: ZeroMQ, Salt-SSH	{salt}: ZeroMQ, Salt-SSH	{salt}: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	{salt}	{salt}	{salt}
Action chains	{salt}	{salt}	{salt}
Staging (pre-download of packages)	{salt}	{salt}	{salt}
Duplicate package reporting	{salt}	{salt}	{salt}
CVE auditing	{question}	{question}	{question}
SCAP auditing	{question}	{question}	{question}
Package verification	{cross}	{cross}	{cross}
Package locking	{cross}	{cross}	{cross}
System locking	{cross}	{cross}	{cross}
System snapshot	{cross}	{cross}	{cross}
Configuration file management	{salt}	{salt}	{salt}

Feature	{ubuntu} 16.04	{ubuntu} 18.04	{ubuntu} 20.04
Package profiles	{salt}: Profiles supported, Sync not supported	{salt}: Profiles supported, Sync not supported	{salt}: Profiles supported, Sync not supported
Power management	{check}	{check}	{check}
Monitoring	{cross}	{salt}	{salt}
Docker buildhost	{question}	{question}	{question}
Build Docker image with OS	{salt}	{salt}	{salt}
Kiwi buildhost	{cross}	{cross}	{cross}
Build Kiwi image with OS	{cross}	{cross}	{cross}

{star} 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 all features of Basic virtual guest management plus 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 {sles} 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 {sles} 15 x86_64 base channel, they can only install or update packages made available through the compatible base channel, or any of its associated child channels.

In the Uyuni {webui} you can browse your available channels by navigating to menu:Software[Channel List > All]. You can modify or create new channels by navigating to menu:Software[Manage > Channels].

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

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

Packages Provided by {phub}

{phub} is an extension to {sle} products that provides additional open source software provided by the {opensuse} community.

IMPORTANT   The packages in {phub} are provided by the {opensuse} community. They are not supported by {suse}.

If you are using {sle} operating systems on your clients, you can enable the {phub} extension to access these additional packages. This provides the {phub} channels, which you can subscribe your clients to.

{phub} provides a large number of packages, which can take a long time to synchronize and consume a large amount of disk space. Do not enable {phub} unless you require the packages it provides.

To avoid unintentionally installing or updating unsupported packages, we recommend that you implement a content lifecycle management strategy that initially denies all {phub} packages. You can then explicitly enable the specific packages you require. For more information about content lifecycle management, see **[**

Administration › Content-lifecycle ›].

Packages Provided by AppStream

For {redhat} based clients, additional packages are available through AppStream. In most cases, the AppStream packages are required to ensure that you have all the software you need.

When you are managing AppStream packages in the Uyuni {webui}, you might notice that you see contradicting suggestions for package updates. This is due to the Uyuni not being able to interpret the modular metadata correctly. You can use the content lifecycle management (CLM) AppStream filter to transform AppStream repositories into non-modular repositories for use with some upgrade operations. For more information about the CLM AppStream filters, see **[Administration › Content-lifecycle-examples ›].**

Packages Provided by EPEL

For {redhat} based clients, additional packages are available through EPEL (extra packages for enterprise Linux). EPEL is an optional package repository that provides additional software.

> **IMPORTANT** The packages in EPEL are provided by the Fedora community. They are not supported by {suse}.

If you are using {redhat} operating systems on your clients, you can enable the EPEL extension to access these additional packages. This provides the EPEL channels, which you can subscribe your clients to.

EPEL provides a large number of packages, which can take a long time to synchronize and consume a large amount of disk space. Do not enable the EPEL repositories unless you require the packages it provides.

To avoid unintentionally installing or updating unsupported packages, we recommended that you implement a content lifecycle management (CLM) strategy that initially denies all EPEL packages. You can then explicitly enable the specific packages you require. For more information about content lifecycle management, see **[Administration › Content-lifecycle ›].**

Unified Installer Updates Channels on {sle} Clients

This channel is used by the unified installer to ensure it is up to date before it installs the operating system. All {sle} products should have access to the installer updates channel during installation.

For {sles} clients the installer updates channel is synchronized by default when you add a product that contains them, and are enabled when you create an autoinstallable distribution with these product channels.

For all other {sle} variants, including {sle} for SAP, you must add the installer updates channel manually. To do this, clone the appropriate {sles} installer updates channel below the base channel of these {sle} variants. When creating an autoinstallable distribution for these {sle} variants after the channel was cloned, it is used automatically.

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 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 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, to provide packages that are not supplied by Uyuni channels.

WARNING	In most cases, client systems do not require local repositories. Local repositories can lead to problems knowing which packages are available on the client. This can lead to installing unexpected packages.

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

Table 15. Local Repository Locations

Client Operating System	Local Repository Directory
{sles}	`/etc/zypp/repos.d`
{opensuse}	`/etc/zypp/repos.d`
{sles} {es}	`/etc/yum.repos.d/`
{rhel}	`/etc/yum.repos.d/`
{centos}	`/etc/yum.repos.d/`
{ubuntu}	`/etc/apt/sources.list.d/`
{debian}	`/etc/apt/sources.list.d/`

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 {webui} by navigating to menu: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 {webui}, navigate to menu: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 btn:[Add Products] and wait until the products have finished synchronizing.

For more information, see **[Installation › Setup-wizard ›]**.

GPG Keys

Clients use GPG keys to check the authenticity of software packages before they are installed. Only trusted software can be installed on clients.

In most cases, you do not need to adjust the GPG settings to be able to install software on your clients.

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

NOTE	If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

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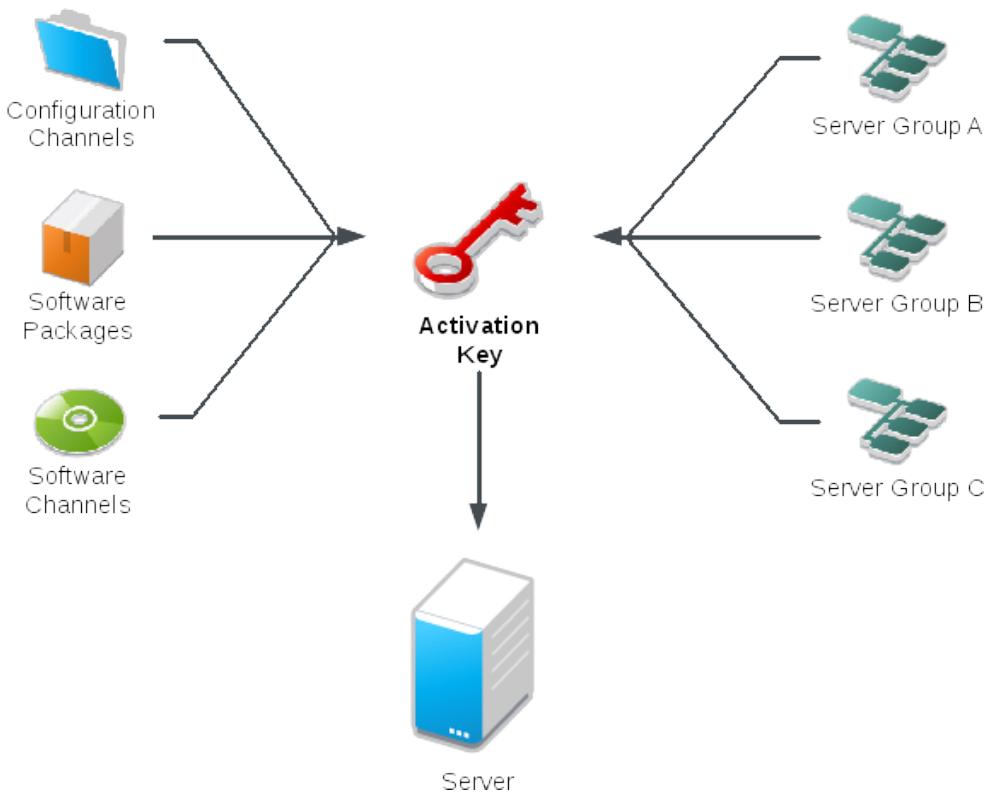
Activation keys are used with traditional and Salt clients to ensure that your clients have the correct software entitlements, are connecting to the appropriate channels, and are subscribed to the relevant groups. Each activation key is bound to an organization, which you can set when you create the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

- Contact method

- Configuration files

- Packages to be installed

- System group assignment

Activation keys are used at the time a client is registered, and not used again. After the client has been registered, the client can be changed in any way, regardless of what the activation key specifies. The association between the activation key and the client is recorded only for historical purposes.

Procedure: Creating an Activation Key

1. In the Uyuni {webui}, as an administrator, navigate to menu:Systems[Activation Keys].

2. Click the btn:[Create Key] button.

3. On the `Activation Key Details` page, in the `Description` field, enter a description of the activation key.

4. In the `Key` field, enter a name for the activation key. For example, `SLES12-SP4` for {sles} 12 SP4.

IMPORTANT	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 {susemgr} tools and updates channels).

7. Check the `Add-On System Types` check box if you need to enable any of the options.

8. We recommend you leave the `Contact Method` set to `Default`.

9. We recommend you leave the `Universal Default` setting unchecked.

10. Click btn:[Create Activation Key] to create the activation key.

11. Check the `Configuration File Deployment` check box to enable configuration management for this key, and click btn:[Update Activation Key] to save this change.

> **NOTE**
> 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 Multiple 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.

> **NOTE**
> Combining activation keys works only on traditional clients. Salt clients do not support combined activation keys. If you use a combined key with a Salt client, only the first key is used.

You can specify multiple activation keys at the command prompt, or in a single autoinstallation profile.

At the command prompt on the Uyuni Server, use the `` `rhnreg_ks `` command, and separate the key names with a comma. To specify multiple keys in a {kickstart} profile, navigate to menu:Systems[Autoinstallation] and edit the profile you want to use.

Be careful when combining activation keys, as conflicts between some values could cause client registration to fail. Check that these values do not have conflicting information before you begin:

- Software packages
- Software child channels
- Configuration channels.

If conflicts are detected, they are handled like this:

- Conflicts in base software channels: registration fails.
- Conflicts in system types: registration fails.
- Conflicts in the `enable configuration` flag: configuration management is enabled.
- If one key is system-specific: registration fails.

Reactivation Keys

Reactivation keys can be used once only to re-register a client and regain all Uyuni settings. Reactivation keys are client-specific, and include the system ID, history, groups, and channels.

To create a reactivation key, navigate to `Systems`, click the client to create a reactivation key for, and

navigate to the menu:Details[Reactivation] tab. Click btn:[Generate New Key] to create the reactivation key. Record the details of the key for later use. Unlike typical activation keys, which are not associated with a specific system ID, keys created here do not show up on the menu:Systems[Activation Keys] page.

For Salt clients, after you have created a reactivation key, you can use it as the `management_key` grain in `/etc/salt/minion.d/susemanager.conf`. For example:

```
grains:
    susemanager:
        management_key: "re-1-daf44db90c0853edbb5db03f2b37986e"
```

Restart the `salt-minion` process to apply the reactivation key.

You can use a reactivation key with a bootstrap script. For more information about bootstrap scripts, see [**Client-configuration › Registration-bootstrap ›**].

For traditional clients, after you have created a reactivation key, you can use it with the `rhnreg_ks` command line utility. This command re-registers the client and restore its Uyuni settings. On traditional clients, you can combine reactivation keys with activation keys to aggregate the settings of multiple keys for a single system profile. For example:

```
rhnreg_ks --server=<server-url>/XMLRPC \
          --activationkey=<reactivation-key>,<activationkey> \
          --force
```

WARNING
If you autoinstall a client with its existing Uyuni profile, the profile uses the reactivation key to re-register the system and restore its settings. Do not regenerate, delete, or use this key while a profile-based autoinstallation is in progress. Doing so causes the autoinstallation to fail.

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 helps you align channel assignments, package installation, system group memberships, and configuration channel assignments. You 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 results in the automatic installation of the corresponding product descriptor package.

- Any package in the `Packages` section is 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 automatically generates 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 makes 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 this server being 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
```

> **IMPORTANT**
>
> 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 are associated with it. Keys should have a specific base channel assigned to them. Using the default base channel is not recommended. For more information, see the client operating system you are installing at [**Client-configuration › Registration-overview ›]**.

Bootstrap Repository

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

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 {webui}, navigate to menu: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 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, existing repositories are updated only with the latest packages. You can configure it to always start with an empty repository instead. To enable this behavior, add or edit this value in `/etc/rhn/rhn.conf`:

```
server.susemanager.bootstrap_repo_flush = 1
```

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 makes 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
```

IMPORTANT On the next update, the new data from {suse} overwrites the original data file, not the new one. You 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 {sle}

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/`.

If you have mirrored more than one product (for example, SLES and SLES for SAP), or if you use custom channels, you might need to specify the parent channel to use when creating the bootstrap repository. This is not required in every situation. For example, some SLES 15 versions have common code bases, so there is no need to specify a parent channel. Use this procedure only if your environment requires it.

OPTIONAL Procedure: Specifying a Parent Channel for a 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 {ibmz} architectures for {slea} use the same bootstrap repository URL at `/srv/www/htdocs/pub/repositories/sle/15/2/bootstrap/`.

When the `flush` option is enabled, and you attempt to generate the bootstrap repository for multiple architectures, only one architecture is 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-SP2-x86_64
mgr-create-bootstrap-repo --no-flush -c SLE-15-SP2-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 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 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 the type of client, and 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 traditional clients. OSAD allows traditional clients to execute scheduled actions immediately. It does not apply to Salt clients.

Contact Methods for Salt Clients

In most cases, Salt clients are registered accurately with the default bootstrap methods.

If you need to use Salt clients in a disconnected setup, you can configure Push via Salt SSH. In this environment, clients are located in a firewall-protected zone called DMZ. For more information about Salt SSH contact method, see [**Client-configuration › Contact-methods-saltssh ›]**.

If you need to manually configure a Salt client to connect to the Uyuni Server, edit the Salt client configuration file with the correct network details. For more information about Salt minion configuration file contact method, see [**Client-configuration › Contact-methods-salt-cfgfile ›]**.

Onboarding Details

Salt has its own database to keep the keys for the minions. This needs to be kept in sync with the Uyuni database. As soon as the key is accepted in Salt, the onboarding process in Uyuni starts. The onboarding process will look for existing systems in the Uyuni database by searching for the `minion_id` and the `machine-id`. If nothing is found, the new system gets created. In case an entry with the `minion_id` or the `machine-id` is found, that system will be migrated to match the new system. In case there is a match for both entries, and they are not the same system, the onboarding will be aborted with an error. In this case the administrator needs to resolve the conflict by removing at least on of the systems.

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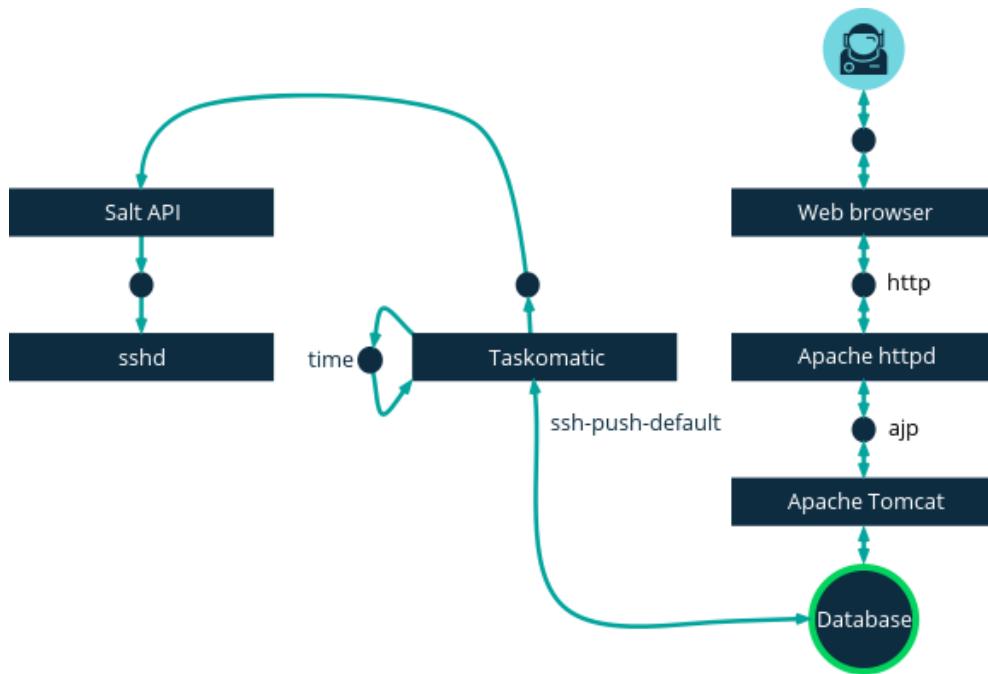
Push via Salt SSH is used in environments where Salt clients cannot reach the Uyuni Server directly. In this environment, clients are located in a firewall-protected zone called a DMZ. No system within the DMZ is authorized to open a connection to the internal network, including the Uyuni Server.

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 the 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.

> **NOTE** {rhel} 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 {webui}, navigate to menu: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 btn:[Bootstrap] to begin registration.

5. Confirm that the system has been registered correctly by navigating to menu: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 menu:Systems[Bootstrapping].

You can also configure persistent parameters that are 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, twenty Salt SSH actions can be executed at a time. You can increase the number of actions that can be executed in parallel, by adding these lines to your configuration file, and adjusting the value of `parallel_threads` upwards. We recommend you keep the number of parallel actions low, to avoid problems:

```
taskomatic.com.redhat.rhn.taskomatic.task.SSHMinionActionExecutor.parallel_threads = <number>
org.quartz.threadPool.threadCount = <value of parallel_threads + 20>
```

This adjusts the number of actions that can run in parallel on any one client and the total number of worker threads used by taskomatic. If actions needs to be run on multiple clients, actions are always executed sequentially on each client.

If the clients are connected through a proxy, you 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 do not work on Salt SSH clients:

- OpenSCAP auditing
- Beacons, resulting in:
  - Installing a package on a system using `zypper` does not invoke the package refresh.
  - Virtual Host functions (for example, a host to guests) does 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/.

Salt Minion Configuration File

In most cases, Salt clients are registered accurately with the default bootstrap methods. However, you can

use Salt to register the client to the Uyuni Server manually by editing the Salt minion file on the client, and providing the fully qualified domain name (FQDN) of the server. This method operates using ports 4505 and 4506 inbound to the server. This method requires no configuration on the Uyuni Server aside from ensuring that these ports are open.

Procedure: Registering Clients with Salt Minion Configuration File

1. On the Salt client, open the `/etc/salt/minion` configuration file.

2. Add or edit this line, using the FQDN of the Uyuni Server:

   ```
   master: <SERVER-FQDN>
   ```

3. Restart the Salt service:

   ```
   service salt-minion restart
   ```

4. On the Uyuni Server, at the command prompt, accept all available Salt clients:

   ```
   salt-key -A
   ```

For more information about the `/etc/salt/minion` configuration file, see https://docs.saltstack.com/en/latest/ref/configuration/minion.html.

Contact Methods for Traditional Clients

Traditional clients can communicate with the Uyuni Server using a range of methods.

The Uyuni daemon (`rhnsd`) runs on traditional client systems and periodically connects with Uyuni to check for new updates and notifications.

Push via SSH is 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 traditional clients. OSAD allows traditional clients 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.
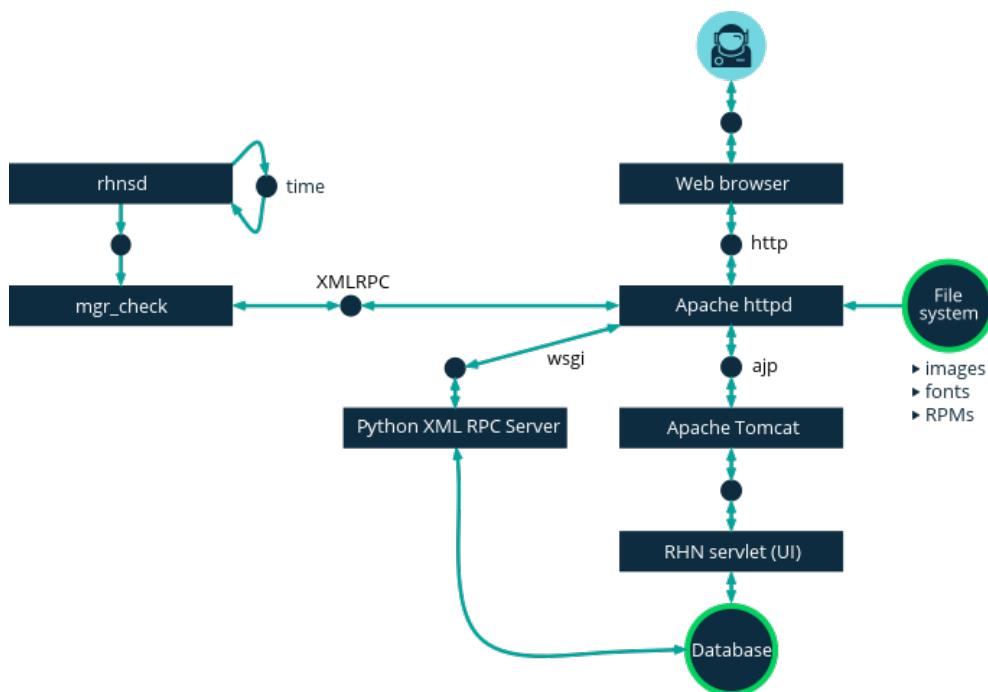
It is only used on {sle} 11 and {rhnminrelease6}, as these systems do not use systemd. On later operating systems, a systemd timer (`rhnsd.timer`) is used and controlled by `rhnsd.service`.

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

By default, it checks 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 Uyuni 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 the 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 changes 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 {sle} 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 is saved as `/etc/systemd/system/rhnsd.timer.d/override.conf`.

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

OSAD

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

> **NOTE** Use OSAD in addition to `rhnsd`. If you disable `rhnsd` your client is 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 uses `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 is 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 then executes 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 fails to connect. Make sure your firewall rules are set to allow the required ports. For more information, see **[Installation › Ports ›]**.

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 conflicts 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`

Component	Location	Path to Configuration File
`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` queues the requests, and refuses 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 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>
```

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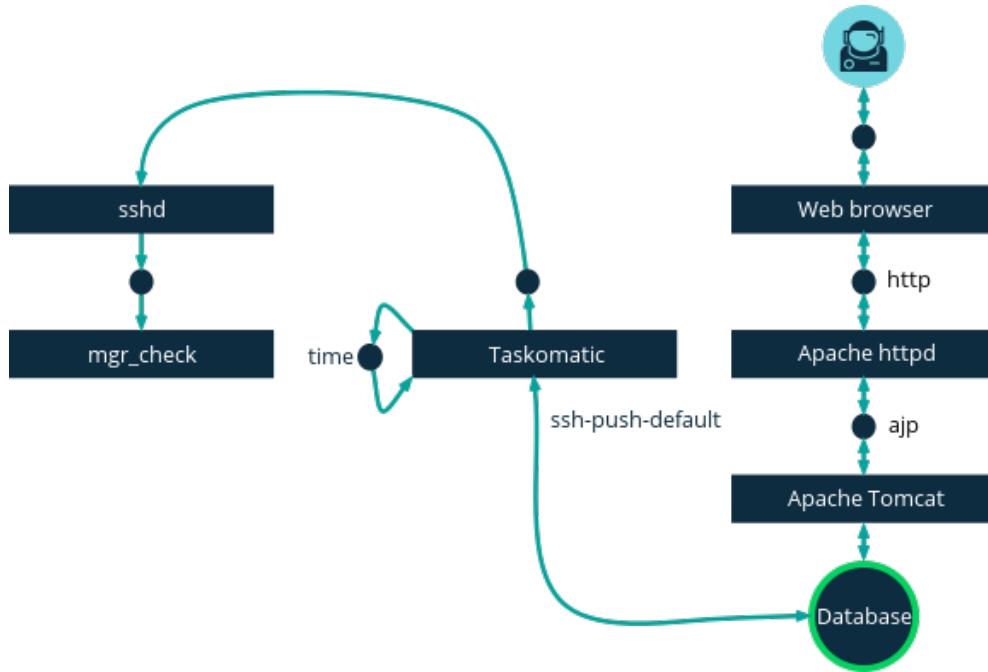
The server uses SSH to contact the clients at regular intervals, checking in and performing scheduled actions and events.

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

IMPORTANT	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 the 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.

3. 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'!
```

4. 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
```

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

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

6. 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 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 needs 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 need to be registered again.

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

Procedure: Registering Clients with Push via SSH

1. 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. OPTIONAL: When you have defined `ssh_push_sudo_user`, you can allow use of the root password by adding the `--notty` option.

3. 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 {webui}, 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 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>
```

Client Registration Overview

There are several ways to register clients to your Uyuni Server. This section covers the various available methods. It also contains information specific to the operating system you intend to run on the client.

Before you begin, check that:

- The client has the date and time synchronized correctly with the Uyuni Server before registration.

- You have created an activation key. For more information about creating activation keys, see [**Client-configuration › Activation-keys ›]**.

IMPORTANT	Do not register the Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [**Large-deployments › Multi-server ›]**.

Client Registration Methods

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

- For Salt clients, we recommend that you register clients using the Uyuni {webui}. For more information, see [**Client-configuration › Registration-webui ›]**.

- If you want more control over the process, have to register many clients, or are registering traditional clients, we recommend that you create a bootstrap script. For more information, see [**Client-configuration › Registration-bootstrap ›]**.

- For Salt clients and even more control over the process, executing single commands on the command line can be useful. For more information, see [**Client-configuration › Registration-cli ›]**.

The client must have the date and time synchronized correctly with the Uyuni Server before registration.

You must create an activation key first, to use bootstrap script or command line method. For more information about creating activation keys, see [**Client-configuration › Activation-keys ›]**.

IMPORTANT	Do not register the Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [**Large-deployments › Multi-server ›]**.

Register Clients with the {webui}

Registering clients with the Uyuni {webui} works for Salt clients only.

<table>
<tr><td>**IMPORTANT**</td><td>Do not register the Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see **[Large-deployments › Multi-server ›]**.</td></tr>
</table>

Procedure: Registering Clients with the {webui}

1. In the Uyuni {webui}, navigate to menu: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. To bootstrap the client with SSH, in the `Authentication` field, check `SSH Private Key`, and upload the SSH private key to use to log in to the client. If your SSH private key requires a passphrase, type it into the `SSH Private Key Passphrase` field, or leave it blank for no passphrase.

6. To bootstrap the client with a password, in the `Authentication` field, check `Password`, and type the password to log in to the client.

7. 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 ›]**.

8. OPTIONAL: In the `Proxy` field, select the proxy to register the client to.

9. 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.

10. OPTIONAL: Check the `Manage System Completely via SSH` checkbox. If you check this option, the client is configured to use SSH for its connection to the server, and no other connection method is configured.

11. Click btn:[Bootstrap] to begin registration.

When the bootstrap process has completed, your client is listed at menu:Systems[System List].

<table>
<tr><td>**IMPORTANT**</td><td>SSH private keys are stored only for the duration of the bootstrapping process. They are deleted from the Uyuni Server as soon as bootstrapping is complete.</td></tr>
</table>

<table>
<tr><td>**WARNING**</td><td>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 {webui}.</td></tr>
</table>

> **WARNING**
> To register and use {centos} 6, {oracle} 6, {rhel} 6, or {sleses} 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 ›].**

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 cannot download the correct packages.

> **NOTE**
> A bootstrap repository is needed for all non-SLE clients, and for SLE clients before version 15. A bootstrap repository includes 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 {webui}, you need to ensure that the client system has Python installed before you begin. For Salt clients running {sles} 12 or older, you also need the `python-xml` package.

> **IMPORTANT**
> *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.

> **NOTE**
> *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. If you register openSUSE Leap 15 or SLE 15 systems using Python 2, the bootstrap script fails.

Create a Bootstrap Script from {webui}

You can use the Uyuni {webui} to create an editable bootstrap script.

Procedure: Creating a Bootstrap Script

1. In the Uyuni {webui}, navigate to menu: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 btn:[Update] to 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
```

IMPORTANT

Do not disable SSL in your bootstrap script. Ensure that `Enable SSL` is checked in the {webui}, 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 ›]**.

WARNING

To register and use {centos} 6, {oracle} 6, {rhel} 6, or {sleses} 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 also need to have trusted GPG keys on your system to install them.

In this procedure, you need to know the exact name of your activation keys. Navigate to menu: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
```

3. Open `bootstrap-sles15.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-sles15
ORG_GPG_KEY=
```

4. 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

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

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

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

```
cat bootstrap-sles15.sh | ssh root@EXAMPLE.COM /bin/bash
```

The script downloads the required dependencies located in the repositories directory you created earlier.

3. When the script has finished running, you can check that your client is registered correctly by opening the Uyuni {webui} and navigating to menu:Systems[Overview] to ensure the new client is listed.

4. If you used the script to register the Salt client, open the Uyuni {webui} and navigate to menu:Salt[Keys] to accept the client key.

WARNING	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 {webui}.

Register on the Command Line (Salt)

In most cases, Salt clients are registered accurately with the default bootstrap methods. However, you can use Salt to register the client to the Uyuni Server manually by editing the Salt minion file on the client, and providing the fully qualified domain name (FQDN) of the server. This method uses ports 4505 and 4506 inbound to the server. This method requires no configuration on the Uyuni Server aside from ensuring that these ports are open.

NOTE	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.

This procedure requires that you have installed the `salt-minion` package on the Salt client before registration. For SLE 12 based clients, you also must have activated the `Advanced Systems Management` module.

Procedure: Registering Clients with Salt Minion Configuration File

1. On the Salt client, open the `minion` configuration file. The configuration file is either located at:

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

   or:

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

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

```
master: SERVER.EXAMPLE.COM
server_id_use_src: adler42
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>'
```

For more information about the `/etc/salt/minion` configuration file, see https://docs.saltstack.com/ en/latest/ref/configuration/minion.html.

> **WARNING**
>
> To register and use {centos} 6, {oracle} 6, {rhel} 6, or {sleses} 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

You can register {sle} and {sleses} clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [**Client-configuration › Activation-keys ›]**.

> **IMPORTANT**
>
> Do not register a Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [**Large-deployments › Multi-server ›]**.

Registering {sle} Clients

This section contains information about registering clients running these {sle} operating systems:

- {sles} 15 SP1
- {sles} 15 SP2
- {sles} 15 SP3

Use the instructions in this chapter for preparing all {sle} products, including {sles} for SAP, {sle} Desktop, {sle} HPC, and {sle} Real Time. You can also use these instructions for older {sle} versions and service packs.

Add Software Channels

Before you register {sle} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The products you need for this procedure are:

Table 16. SLE Products - WebUI

OS Version	Product Name
{sles} 15 SP1	SUSE Linux Enterprise Server 15 SP1 x86_64
{sles} 15 SP2	SUSE Linux Enterprise Server 15 SP2 x86_64
{sles} 15 SP3	SUSE Linux Enterprise Server 15 SP3 x86_64

Procedure: Adding Software Channels

1. In the Uyuni {webui}, navigate to menu: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 btn:[Add Products] and wait until the products have finished synchronizing.

Alternatively, you can add channels at the command prompt. The channels you need for this procedure are:

Table 17. SLE Products - CLI

OS Version	Base Channel
{sles} 15 SP1	sle-product-sles15-sp1-pool-x86_64
{sles} 15 SP2	sle-product-sles15-sp2-pool-x86_64
{sles} 15 SP3	sle-product-sles15-sp3-pool-x86_64

Procedure: Adding Software Channels at the Command Prompt

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

```
mgr-sync add channel <channel_label_1>
mgr-sync add channel <channel_label_2>
mgr-sync add channel <channel_label_n>
```

2. Synchronization starts automatically. If you want to synchronize the channels manually, use:

```
mgr-sync sync --with-children <channel_name>
```

3. Ensure the synchronization is complete before continuing.

To add the client tools, add these channels from the command prompt:

Table 18. SUSE Linux Enterprise Channels - CLI

OS Version	Client Channel
{sles} 15 SP1	sles15-sp1-uyuni-client
{sles} 15 SP2	sles15-sp2-uyuni-client
{sles} 15 SP3	sles15-sp3-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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

IMPORTANT   {sle} channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

> **NOTE**
> If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

> **NOTE**
> Use the same GPG key for both {sles} 15 and {sles} 12 clients. The correct key is called `sle12-gpg-pubkey-39db7c82.key`.

Register Clients

To register your {sle} 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 {sleses} Clients

This section contains information about registering traditional and Salt clients running {sleses} ({es}) operating systems. {es} clients are based on {rhel} or {centos}. They are sometimes also called SLESES, RES or {redhat} {es}.

The {es} software channels provided by {suse} only provide updates to packages, they do not provide the packages themselves. To register {es} clients, you need to create custom empty base and child channels on your Uyuni Server, then import the {redhat} or {centos} packages into the custom channels. You must obtain the initial packages directly from {redhat} or {centos} before you can apply the updates provided by the {es} software channels.

IMPORTANT	You are responsible for arranging access to {redhat} or {centos} base media repositories and installation media.
IMPORTANT	{suse} does not provide support for {es} systems on {uyuni}.
IMPORTANT	Traditional clients are not available on {es} 8. {es} 8 clients are only supported as Salt clients.

Add Software Channels

For {es} clients, some required packages are contained on the {rhel} or {centos} installation media. You must have these packages installed before you can register a {es} client.

The {es} product is provided by {scc}. This also includes the client tools package.

Before you register {es} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

You need to select two different sets of channels, one for {es} and the other for the Client Tools.

You need an activation key associated with the correct {es} channels. For more information about activation keys, see **[Client-configuration › Activation-keys ›]**.

The channels you need for this procedure are:

Table 19. ES Channels - CLI

OS Version	Base Channel	Client Channel	Tools Channel
{es} 6	rhel-x86_64-server-6	-	res6-suse-manager-tools-x86_64
{es} 7	rhel-x86_64-server-7	-	res7-suse-manager-tools-x86_64
{es} 8	rhel-x86_64-server-8	-	res8-suse-manager-tools-x86_64

NOTE	{es} 6 is now at end-of-life, and the ISO images provided in the repository are out of date. Bootstrapping new {es} 6 clients using these packages will fail. If you need to bootstrap new {es} 6 clients, follow the troubleshooting procedure in **[Client-configuration › Tshoot-clients ›]**.

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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

IMPORTANT	The AppStream repository provides modular packages. This results in the Uyuni {webui} showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the {webui} or API. You can use the AppStream filter with content lifecycle management (CLM) to transform modular repositories into regular repositories. Make sure to include `python:3.6` using an AppStream filter if you want to use `spacecmd` on the clients. Alternatively, you can use Salt states to manage modular packages on Salt clients, or use the `dnf` command on the client. For more information about CLM, see **[Administration › Content-lifecycle ›]**.

Add Base Media

The base {es} channel does not contain any packages, because {suse} does not provide {rhel} or {centos} base media. You need to obtain base media from {redhat} or {centos}, which you can add as a child channel to the {es} 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 {rhel} or {centos} media. All packages on the base media must be mirrored into a child channel.

You can freely choose the names for the channels.

Procedure: Creating Custom Channels

1. On the Uyuni Server {webui}, navigate to menu:Software[Manage > Channels].

2. Click btn:[Create Channel] and set the appropriate parameters for the channels.

3. In the `Parent Channel` field, select the appropriate base channel.

4. Click btn:[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 menu:Software[Channel List > All].

IMPORTANT	For {redhat} 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

If you are using modular channels, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the `spacecmd` package will fail.

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>
```

4. Import the packages into the child channel you created earlier:

```
spacewalk-repo-sync -c <channel-label> -u
file:///srv/www/htdocs/pub/<os_name>/<repopath>/
```

OPTIONAL: Add Base Media from a Content URL

Alternatively, if you have access to a content URL provided by {redhat} CDN or {centos}, you can create a custom repository to mirror the packages.

The details you need for this procedure are:

Table 20. ES Custom Repository Settings

Option	Parameter
Repository URL	The content URL provided by {redhat} CDN or {centos}
Has Signed Metadata?	Uncheck all {redhat} Enterprise repositories
SSL CA Certificate	`redhat-uep` ({redhat} only)
SSL Client Certificate	`Entitlement-Cert-date` ({redhat} only)
SSL Client Key	`Entitlement-Key-date` ({redhat} only)

Procedure: Creating Custom Repositories

1. On the Uyuni Server {webui}, navigate to menu:Software[Manage > Repositories].

2. Click btn:[Create Repository] and set the appropriate parameters for the repository.

3. Click btn:[Create Repository].

4. Repeat for all repositories you need to create.

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 {webui}, navigate to menu: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 btn:[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 btn:[Sync Now] to begin synchronization immediately.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

> **NOTE**
>
> The {es} 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 {es} Clients

Your {es} clients are now ready to be registered.

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

> **WARNING**
>
> To register and use {sleses} 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 ›]**.

openSUSE Client Registration

You can register {opensuse} clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see **[Client-configuration › Activation-keys ›]**.

> **IMPORTANT**
>
> Do not register a Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see **[Large-deployments › Multi-server ›]**.

Registering openSUSE Leap 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, you need to add the required software channels, and synchronize them.

The products you need for this procedure are:

Table 21. OpenSUSE Channels - CLI

OS Version	Base Channel	Client Channel	Updates Channel	Non-OSS Channel	Non-OSS Updates Channel
{opensuse} Leap 15.1	opensuse_leap15_1	opensuse_leap15_1-uyuni-client	opensuse_leap15_1-updates	opensuse_leap15_1-non-oss	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	opensuse_leap15_2-non-oss-updates

Table 22. OpenSUSE Channels - CLI

OS Version	Base Channel	Client Channel	Updates Channel	Non-OSS Channel	Non-OSS Updates Channel	Backports Updates Channel	SLE Updates Channel
{opensuse} Leap 15.3	opensuse_leap15_3	opensuse_leap15_3-uyuni-client	opensuse_leap15_3-updates	opensuse_leap15_3-non-oss	opensuse_leap15_2-non-oss-updatess	opensuse_leap15_3-backports-update	opensuse_leap15_3-sle-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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

> **IMPORTANT**
>
> {opensuse} channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

> **NOTE**
>
> If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

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
```

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

Registering {micro} Clients

This section contains information about registering clients running these {micro} operating systems:

- {micro} 5.0 {x86_64}

- {micro} 5.0 {arm64}

WARNING

Support for {micro} clients is provided as a technology preview for testing purposes, and not all features are fully functional at this stage. This feature is expected to be fully supported in a later version of Uyuni. Do not use this feature on production systems.

The {micro} is an ultra-reliable, lightweight operating system purpose built for edge computing. It leverages the enterprise hardened security and compliance components of SUSE Linux Enterprise and merges them with a modern, immutable, developer-friendly OS platform.

The {micro} uses transactional updates. Transactional updates are atomic (all updates are applied only if all updates succeed) and support rollbacks. They do not affect a running system as no changes are activated until after the system is rebooted. This information is displayed in the menu:Systems[Details > Overview] subtab.

For more information on transactional updates, see https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-transactional-updates.html

Add Software Channels

Before you register {micro} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

Register Clients

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

Registering {opensuse} {microos} Clients

This section contains information about registering clients running these {opensuse} {microos} operating systems:

- {opensuse} {microos}

> **WARNING**
>
> Support for {opensuse} {microos} clients is provided as a technology preview for testing purposes, and not all features are fully functional at this stage. This feature is expected to be fully supported in a later version of Uyuni. Do not use this feature on production systems.

Add Software Channels

Before you register {opensuse} {microos} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 23. openSUSE MicroOS Channels - CLI

OS Version	Base Channel	Client Channel	Updates Channel
{opensuse} {microos}	opensuse_tumbleweed	opensuse_tumbleweed-non-oss	opensuse_tumbleweed-update

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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

> **IMPORTANT** {opensuse} {microos} channels can be very large. Synchronization can sometimes take several hours.

Trust Certificates Keys on Clients

{opensuse} {microos} is not yet fully enabled, so there are some manual steps to trust the Uyuni SSL certificate on {microos} clients.

Procedure: Installing and Configuring Salt

1. On the client, at the command prompt, as root, retrieve the SSL certificate file from the server:

```
curl -k https://uyuni-server.hispa-net.com/pub/RHN-ORG-TRUSTED-SSL-CERT -o
/etc/pki/trust/anchors/RHN-ORG-TRUSTED-SSL-CERT
```

2. Update the certificates on the client:

```
update-ca-certificates
```

3. Install the required packages:

```
transactional-update pkg install salt-minion dmidecode
```

4. Reboot the client. If a message is shown indicating that there is a conflict with busybox-hostname, click btn:[Deinstallation of busybox-hostname].

5. Create a new file called /etc/salt/minion.d/susemanager-transactional.conf with this content:

```
module_executors:
 - transactional_update
 - direct_call
```

Your Uyuni Server will not show the true state of the client in the {webui} until after you have rebooted the client. This feature is expected to be fully supported in a later version of Uyuni.

> **NOTE** If Salt is failing to install any software, you could be using an old version of Salt. Upgrade your Salt packages to the latest version to resolve this problem.

Register Clients

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

{alibabaclo} Client Registration

You can register {alibabaclo} clients to your Uyuni Server. The method and details vary depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

> **NOTE** Some {alibabaclo} 2 instances will need two tries to register successfully.

You must also have created an activation key. For more information about creating activation keys, see [**Client-configuration › Activation-keys ›**].

Registering {alibabaclo} Clients

This section contains information about registering traditional and Salt clients running {alibabaclo} operating systems.

IMPORTANT	The traditional stack is available on {alibabaclo} 2 but it is not supported. {alibabaclo} 2 clients are only supported as Salt clients.	

WARNING	Some {alibabaclo} 2 instances will need two tries to register successfully.	

Add Software Channels

Before you register {alibabaclo} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 24. Alibaba Cloud Linux Channels - CLI

OS Version	Core Channel	Updates Channel	Client Channel
{alibabaclo} 2	alibaba-2	alibaba-2-updates	alibaba-2-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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

NOTE	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 {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

Create an Activation Key

You need to create an activation key that is associated with your {alibabaclo} channels.

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

Register Clients

{alibabaclo} clients are registered in the same way as all other clients.

Some {alibabaclo} 2 instances will fail to register on the first try.

This is due to a to a known bug in the {alibabaclo} 2 image.

The `python-urlgrabber3` package is provided both as a Python pip package and an RPM package, which can cause a conflict on the first attempt to register.

If your instance is based on one of the affected image versions, the client should register correctly on the second registration attempt.

For more information about client registration, see **[Client-configuration › Registration-overview ›]**.

{almalinux} Client Registration

You can register {almalinux} clients to your Uyuni Server. The method and details vary depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see **[Client-configuration › Activation-keys ›]**.

Registering {almalinux} Clients

This section contains information about registering Salt clients running {almalinux} operating systems.

IMPORTANT	Traditional clients are not available on {almalinux} 8. {almalinux} 8 clients are only supported as Salt clients.

WARNING	When created at AWS, {almalinux} instances always have the same `machine-id` id at `/etc/machine-id`. Make sure you regenerate the `machine-id` after the instance is created. For more information, see [**Administration › Tshoot-registerclones ›]**.

Add Software Channels

NOTE	Registering {almalinux} clients to Uyuni is tested with the default SELinux configuration of `enforcing` with a `targeted` policy. You do not need to disable SELinux to register {almalinux} clients to Uyuni.

Before you can register {almalinux} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 25. CentOS Channels - CLI

OS Version	Base Channel	Client Channel	AppStream Channel
{almalinux} 8	almalinux8	almalinux8-uyuni-client	almalinux8-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. Ensure you specify the correct architecture:

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

2. Synchronize the channels:

   ```
   spacewalk-repo-sync -p <base_channel_label>
   ```

3. Ensure the synchronization is complete before continuing.

NOTE	The client tools channel provided by `spacewalk-common-channels` is sourced from {uyuni} and not from {suse}.

IMPORTANT	For {almalinux} 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

If you are using modular channels, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the `spacecmd` package will fail.

NOTE	You might notice some disparity in the number of packages available in the AppStream channel between upstream and the Uyuni channel. You might also see different numbers if you compare the same channel added at a different point in time. This is due to the way that {almalinux} manages their repositories. {almalinux} removes older version of packages when a new version is released, while Uyuni keeps all of them, regardless of age.

IMPORTANT	The AppStream repository provides modular packages. This results in the Uyuni {webui} showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the {webui} or API.
	You can use the AppStream filter with content lifecycle management (CLM) to transform modular repositories into regular repositories. Make sure to include `python:3.6` using an AppStream filter if you want to use `spacecmd` on the clients.
	Alternatively, you can use Salt states to manage modular packages on Salt clients, or use the `dnf` command on the client. For more information about CLM, see **[Administration › Content-lifecycle ›]**.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

Create an Activation Key

You need to create an activation key that is associated with your {almalinux} channels.

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

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

> **NOTE**
> If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

Register Clients

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

Manage Errata

When you update {almalinux} clients, the packages include metadata about the updates.

{amazon} Client Registration

You can register {amazon} clients to your Uyuni Server. The method and details vary depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see **[Client-configuration › Activation-keys ›]**.

Registering {amazon} Clients

This section contains information about registering traditional and Salt clients running {amazon} operating systems.

IMPORTANT	Traditional clients are not available on {amazon} 2. {amazon} 2 clients are only supported as Salt clients.

WARNING	When created at AWS, {amazon} instances always have the same `machine-id` id at `/etc/machine-id`. Make sure you regenerate the `machine-id` after the instance is created. For more information, see **[Administration › Tshoot-registerclones ›]**.

Add Software Channels

Before you register {amazon} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 26. Amazon Linux Channels - CLI

OS Version	**Core Channel**	**Client Channel**
{amazon} 2	amazonlinux2-core	amazonlinux2-uyuni-client

NOTE	Make sure you also add and sync `amazonlinux2-extra-docker` channel if you plan to use Docker at your {amazon} instances.

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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

NOTE 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 {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

Create an Activation Key

You need to create an activation key that is associated with your {amazon} channels.

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

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

NOTE If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

Register Clients

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

CentOS Client Registration

You can register {centos} clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see **[Client-configuration › Activation-keys ›]**.

Registering {centos} Clients

This section contains information about registering traditional and Salt clients running {centos} operating systems.

IMPORTANT	{centos} clients are based on {centos} and are unrelated to {sleses}, RES, {redhat}, or {es}. 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.

NOTE	Traditional clients are not available on {centos} 8. {centos} 8 clients are only supported as Salt clients.

NOTE	Registering {centos} clients to Uyuni is tested with the default SELinux configuration of `enforcing` with a `targeted` policy. You do not need to disable SELinux to register {centos} clients to Uyuni.

Add Software Channels

Before you can register {centos} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 27. CentOS Channels - CLI

OS Version	Base Channel	Client Channel	Updates/Appstream 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

> **NOTE** {centos} 6 is now at end-of-life, and the ISO images provided in the repository are out of date. Bootstrapping new {centos} 6 clients using these packages will fail. If you need to bootstrap new {centos} 6 clients, follow the troubleshooting procedure in [**Client-configuration › Tshoot-clients ›].**

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. Ensure you specify the correct architecture:

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

2. Synchronize the channels:

   ```
   spacewalk-repo-sync -p <base_channel_label>
   ```

3. Ensure the synchronization is complete before continuing.

> **NOTE** The client tools channel provided by `spacewalk-common-channels` is sourced from {uyuni} and not from {suse}.

> **IMPORTANT** For {centos} 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

If you are using modular channels, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the `spacecmd` package will fail.

> **NOTE**
>
> You might notice some disparity in the number of packages available in the AppStream channel between upstream and the Uyuni channel. You might also see different numbers if you compare the same channel added at a different point in time. This is due to the way that {centos} manages their repositories. {centos} removes older version of packages when a new version is released, while Uyuni keeps all of them, regardless of age.

> **IMPORTANT**
>
> The AppStream repository provides modular packages. This results in the Uyuni {webui} showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the {webui} or API.
>
> You can use the AppStream filter with content lifecycle management (CLM) to transform modular repositories into regular repositories. Make sure to include `python:3.6` using an AppStream filter if you want to use `spacecmd` on the clients.
>
> Alternatively, you can use Salt states to manage modular packages on Salt clients, or use the `dnf` command on the client. For more information about CLM, see **[Administration › Content-lifecycle ›]**.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

Create an Activation Key

You 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, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

> **NOTE**
>
> If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

Register Clients

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

> **WARNING**
>
> 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.

WARNING

The authors of CEFS provide patches or errata on a best-effort basis, in the hope they are useful but with no guarantees of correctness or currency. This could mean that the patch dates could be incorrect, and in at least one case, the published data was shown to be more than a month old. For more information on these cases, see https://github.com/stevemeier/cefs/issues/28#issuecomment-656579382 and https://github.com/stevemeier/cefs/issues/28#issuecomment-656573607.

Any problems or delays with the patch data might result in unreliable patch information being imported to your Uyuni Server. This would cause reports, audits, CVE updates, or other patch-related information to also be incorrect. Please consider alternatives to using this service, such as independently verifying patch data, or choosing a different operating system, depending on your security-related requirements and certifications criteria.

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 errata service dependencies:

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

3. Add or edit this line in `/etc/rhn/rhn.conf`:

   ```
   java.allow_adding_patches_via_api = centos7-updates-x86_64,centos7-x86_64,centos7-extras-x86_64
   ```

4. Restart Tomcat:

```
systemctl restart tomcat
```

1. Create a file for your errata script:

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

2. 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-updates-x86_64,centos7-x86_64,centos7-extras-x86_64 \
--publish --rhsa-oval /usr/local/centos/com.redhat.rhsa-RHEL7.xml
```

3. 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/.

Debian Client Registration

You can register {debian} clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see **[Client-configuration › Activation-keys ›]**.

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

Registering {debian} Clients

This section contains information about registering Salt clients running {debian} operating systems.

IMPORTANT	{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.

NOTE	{debian} is supported for Salt clients only. Traditional clients are not supported.

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 required packages on the client before you attempt to register. On the client, at the command prompt, as root, run:

```
apt install apt-transport-https python-apt python3-apt
```

- 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 with the Uyuni Server before registration.

Add Software Channels

Before you can register {debian} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 28. 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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

> **NOTE** {debian} channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

> **NOTE** If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

> **NOTE** {debian} clients can require multiple GPG keys to be installed.

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
```

For {debian} 10, select `debian10-amd64-uyuni` when prompted.

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

Oracle Client Registration

You can register {oracle} clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see **[Client-configuration › Activation-keys ›]**.

Registering {oracle} Clients

This section contains information about registering traditional and Salt clients running {oracle} operating systems.

> **IMPORTANT** Traditional clients are not available on {oracle} 8. {oracle} 8 clients are only supported as Salt clients.

Add Software Channels

Before you register {oracle} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 29. Oracle Channels - CLI

OS Version	Base Channel	Client Channel	Updates Channel
{oracle} 6	oraclelinux6	oraclelinux6-uyuni-client	-
{oracle} 7	oraclelinux7	oraclelinux7-uyuni-client	-
{oracle} 8	oraclelinux8	oraclelinux8-uyuni-client	oraclelinux8-appstream

> **NOTE**
>
> {oracle} 6 is now at end-of-life, and the ISO images provided in the repository are out of date. Bootstrapping new {oracle} 6 clients using these packages will fail. If you need to bootstrap new {oracle} 6 clients, follow the troubleshooting procedure in [**Client-configuration › Tshoot-clients ›]**.

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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

> **NOTE**
>
> The client tools channel provided by `spacewalk-common-channels` is sourced from {uyuni} and not from {suse}.

> **IMPORTANT**
>
> For {oracle} 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

If you are using modular channels, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the `spacecmd` package will fail.

IMPORTANT	The AppStream repository provides modular packages. This results in the Uyuni {webui} showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the {webui} or API.

You can use the AppStream filter with content lifecycle management (CLM) to transform modular repositories into regular repositories. Make sure to include `python:3.6` using an AppStream filter if you want to use `spacecmd` on the clients.

Alternatively, you can use Salt states to manage modular packages on Salt clients, or use the `dnf` command on the client. For more information about CLM, see **[Administration › Content-lifecycle ›]**. |

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

Create an Activation Key

You need to create an activation key that is associated with your {oracle} channels.

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

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

NOTE	If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see [**Salt › Custom-states ›].**

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 channel assigned 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.

Register Clients

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

WARNING	To register and use {oracle} 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

You can register {rhel} clients to your Uyuni Server using either the {redhat} content delivery network (CDN), or {redhat} update infrastructure (RHUI). The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see [**Client-configuration › Activation-keys ›].**

Registering {rhel} Clients with CDN

If you are running {rhel} clients directly, rather than using {sleses}, you need to use Red Hat sources to retrieve and update packages. This section contains information about using the {redhat} content delivery network (CDN) to register traditional and Salt clients running {rhel} operating systems.

For information about using {redhat} update infrastructure (RHUI) instead, see [**Client-configuration ›**

Clients-rh-rhui ›].

IMPORTANT	{rhel} clients are based on {redhat} and are unrelated to {sleses}, RES, or {sles}. You are responsible for arranging access to {redhat} base media repositories and {rhela} installation media, as well as connecting Uyuni Server to the {redhat} content delivery network. You must obtain support from {redhat} for all your {rhela} systems. If you do not do this, you might be violating your terms with {redhat}.

NOTE	Traditional clients are available on {rhel} 6 and 7 only. {rhel} 8 clients are supported as Salt clients.

Import Entitlements and Certificates

{redhat} clients require a {redhat} 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 need to repeat this process at the end of every support subscription period.

{redhat} supplies 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.

{redhat} clients use a URL to replicate repositories. The URL changes depending on where the {redhat} client is registered.

{redhat} clients can be registered in three different ways:

- {redhat} content delivery network (CDN) at redhat.com

- {redhat} Satellite Server

- {redhat} update infrastructure (RHUI) in the cloud

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

For information about using {redhat} update infrastructure (RHUI) instead, see **[Client-configuration** › **Clients-rh-rhui** › **]**.

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

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

{redhat} 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 {redhat} 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/
```

> **NOTE**
> Your entitlement certificate and key both have a file extension of `.pem`. The key also has `key` in the filename.

3. Copy the {redhat} 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 {redhat} 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 {redhat} certificate.

Procedure: Importing Certificates to the Server

1. On the Uyuni Server {webui}, navigate to menu:Systems[Autoinstallation > GPG and SSL Keys].

2. Click btn:[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 btn:[Create Key].

4. Click btn:[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 btn:[Create Key].

6. Click btn:[Create Stored Key/Cert] and set these parameters for the {redhat} 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 {redhat} certificate, and select the certificate file.

7. Click btn:[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.

> **IMPORTANT** You can only create custom versions of {redhat} repositories if you have the correct entitlements in your {redhat} Portal.

The details you need for this procedure are:

Table 30. Red Hat Custom Repository Settings

Option	Setting
Repository URL	The content URL provided by {redhat} CDN
Has Signed Metadata?	Uncheck all {redhat} 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 {webui}, navigate to menu:Software[Manage > Repositories].

2. Click btn:[Create Repository] and set the appropriate parameters for the repository.

3. Click btn:[Create Repository].

4. Repeat for all repositories you need to create.

The channels you need for this procedure are:

Table 31. Red Hat Custom Channels

OS Version	Base Product	Base Channel
{redhat} 6	RHEL6 Base x86_64	rhel6-pool-x86_64
{redhat} 7	RHEL7 Base x86_64	rhel7-pool-x86_64
{redhat} 8	RHEL or SLES ES or CentOS 8 Base	rhel8-pool-x86_64

> **NOTE**
>
> {redhat} 6 is now at end-of-life, and the ISO images provided in the repository are out of date. Bootstrapping new {redhat} 6 clients using these packages will fail. If you need to bootstrap new {redhat} 6 clients, follow the troubleshooting procedure in **[Client-configuration › Tshoot-clients ›]**.

Procedure: Creating Custom Channels

1. On the Uyuni Server {webui}, navigate to menu:Software[Manage > Channels].

2. Click btn:[Create Channel] and set the appropriate parameters for the channels.

3. In the `Parent Channel` field, select the appropriate base channel.

4. Click btn:[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 menu:Software[Channel List > All].

> **IMPORTANT**
>
> For {redhat} 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

If you are using modular channels, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the `spacecmd` package will fail.

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 {webui}, navigate to menu: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 btn:[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 btn:[Sync Now] to begin synchronization immediately.

Add Software Channels

Before you register {redhat} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 32. Red Hat Channels - CLI

OS Version	Base Channel	Client Channel	Tools Channel
{redhat} 6	rhel-x86_64-server-6	-	res6-suse-manager-tools-x86_64
{redhat} 7	rhel-x86_64-server-7	-	res7-suse-manager-tools-x86_64
{redhat} 8	rhel-x86_64-server-8	-	res8-suse-manager-tools-x86_64

> **NOTE**
>
> {redhat} 6 is now at end-of-life, and the ISO images provided in the repository are out of date. Bootstrapping new {redhat} 6 clients using these packages will fail. If you need to bootstrap new {redhat} 6 clients, follow the troubleshooting procedure in **[Client-configuration › Tshoot-clients ›]**.

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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

> **NOTE**
>
> The client tools channel provided by `spacewalk-common-channels` is sourced from {uyuni} and not from {suse}.

> **IMPORTANT**
>
> The AppStream repository provides modular packages. This results in the Uyuni {webui} showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the {webui} or API.
>
> You can use the AppStream filter with content lifecycle management (CLM) to transform modular repositories into regular repositories. Make sure to include `python:3.6` using an AppStream filter if you want to use `spacecmd` on the clients.
>
> Alternatively, you can use Salt states to manage modular packages on Salt clients, or use the `dnf` command on the client. For more information about CLM, see **[Administration › Content-lifecycle ›].**

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

> **NOTE**
>
> {rhel} 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 {webui}, navigate to menu:Configuration[Channels].

2. Click btn:[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 btn:[Create Config Channel]

4. Click btn:[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 btn:[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 btn:[Update Configuration File].

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

Procedure: Creating a System Group for {rhel} Clients

1. On the Uyuni Server {webui}, navigate to menu:Systems[System Groups].

2. Click btn:[Create Group].

   ◦ In the Name field, type rhel-systems.

   ◦ In the Description field, type All RHEL systems.

3. Click btn:[Create Group].

4. Click `States` tab.

5. Click `Configuration Channels` tab.

6. Type `subscription-manager: disable yum plugins` at the search box.

7. Click btn:[Search] to see the state.

8. Click the checkbox for the state at the `Assign` column.

9. Click btn:[Save changes].

10. Click btn:[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 {webui}, navigate to menu: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 btn:[Join Selected Groups].

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

> **NOTE**
>
> If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

Register Clients

To register your {redhat} 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 ›]**.

WARNING To register and use {rhel} 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 ›]**.

Registering {rhel} Clients with RHUI

If you are running {rhel} clients directly, rather than using {sleses}, you need to use Red Hat sources to retrieve and update packages. This section contains information about using {redhat} update infrastructure (RHUI) to register traditional and Salt clients running {rhel} 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 {redhat} content delivery network (CDN) to manage your {rhel} subscriptions. For information about using {redhat} CDN, see **[Client-configuration › Clients-rh-cdn ›]**.

IMPORTANT {rhel} clients are based on {redhat} and are unrelated to {sleses}, RES, or {sles}. You are responsible for connecting Uyuni Server to the {redhat} update infrastructure. All clients that get updates using this RHUI certificate need to be correctly licensed, please check with your cloud provider and the {redhat} terms of service for more information.

NOTE When {rhel} clients registered with RHUI are switched off, {redhat} might declare the certificate invalid. In this case, you need to turn the client on again, or get a new RHUI certificate.

> **NOTE** Traditional clients are available on {rhel} 6 and 7 only. {rhel} 8 clients are supported as Salt clients.

Import Entitlements and Certificates

{redhat} clients require a {redhat} certificate authority (CA) and entitlement certificate, and an entitlement key.

{redhat} clients use a URL to replicate repositories. The URL changes depending on where the {redhat} client is registered.

{redhat} clients can be registered in three different ways:

- {redhat} content delivery network (CDN) at redhat.com

- {redhat} Satellite Server

- {redhat} update infrastructure (RHUI) in the cloud

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

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

> **IMPORTANT** 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.

The keys and certificates might have slightly different names to those shown here. Your entitlement certificate and the {redhat} 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. Amazon EC2:

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

Google Cloud Platform:

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

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

Amazon EC2:

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

Google Cloud Platform:

```
cp /etc/pki/rhui/ca.crt /<example>/entitlement
```

To manage repositories on your {redhat} 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 {redhat} certificate.

Procedure: Importing Certificates to the Server

1. On the Uyuni Server {webui}, navigate to menu:Systems[Autoinstallation > GPG and SSL Keys].

2. Click btn:[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 btn:[Create Key].

4. Click btn:[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 btn:[Create Key].

6. Click btn:[Create Stored Key/Cert] and set these parameters for the {redhat} 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 {redhat} certificate, and select the certificate file.

7. Click btn:[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.

> **IMPORTANT** You can only create custom versions of {redhat} repositories if you have the correct entitlements in your {redhat} Portal.

The details you need for this procedure are:

Table 33. Red Hat Custom Repository Settings

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

Procedure: Creating Custom Repositories

1. On the Uyuni Server {webui}, navigate to menu:Software[Manage > Repositories].

2. Click btn:[Create Repository] and set the appropriate parameters for the repository.

3. Click btn:[Create Repository].

4. Repeat for all repositories you need to create.

The channels you need for this procedure are:

Table 34. Red Hat Custom Channels

OS Version	Base Product	Base Channel
{redhat} 6	RHEL6 Base x86_64	rhel6-pool-x86_64
{redhat} 7	RHEL7 Base x86_64	rhel7-pool-x86_64

OS Version	Base Product	Base Channel
{redhat} 8	RHEL or SLES ES or CentOS 8 Base	rhel8-pool-x86_64

NOTE

{redhat} 6 is now at end-of-life, and the ISO images provided in the repository are out of date. Bootstrapping new {redhat} 6 clients using these packages will fail. If you need to bootstrap new {redhat} 6 clients, follow the troubleshooting procedure in **[Client-configuration › Tshoot-clients ›]**.

Procedure: Creating Custom Channels

1. On the Uyuni Server {webui}, navigate to menu:Software[Manage > Channels].

2. Click btn:[Create Channel] and set the appropriate parameters for the channels.

3. In the `Parent Channel` field, select the appropriate base channel.

4. Click btn:[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 menu:Software[Channel List > All].

IMPORTANT

For {redhat} 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot 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 {webui}, navigate to menu: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 btn:[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 btn:[Sync Now] to begin synchronization immediately.

Add Software Channels

Before you register {redhat} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 35. Red Hat Channels - CLI

OS Version	Base Channel	Client Channel	Tools Channel
{redhat} 6	rhel-x86_64-server-6	-	res6-suse-manager-tools-x86_64
{redhat} 7	rhel-x86_64-server-7	-	res7-suse-manager-tools-x86_64
{redhat} 8	rhel-x86_64-server-8	-	res8-suse-manager-tools-x86_64

> **NOTE**
> {redhat} 6 is now at end-of-life, and the ISO images provided in the repository are out of date. Bootstrapping new {redhat} 6 clients using these packages will fail. If you need to bootstrap new {redhat} 6 clients, follow the troubleshooting procedure in [**Client-configuration › Tshoot-clients ›]**.

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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

> **NOTE**
> The client tools channel provided by `spacewalk-common-channels` is sourced from {uyuni} and not from {suse}.

IMPORTANT	The AppStream repository provides modular packages. This results in the Uyuni {webui} showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the {webui} or API. You can use the AppStream filter with content lifecycle management (CLM) to transform modular repositories into regular repositories. Make sure to include `python:3.6` using an AppStream filter if you want to use `spacecmd` on the clients. Alternatively, you can use Salt states to manage modular packages on Salt clients, or use the `dnf` command on the client. For more information about CLM, see **[Administration › Content-lifecycle ›]**.

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 {redhat} 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 {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

NOTE {rhel} channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

NOTE If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›].**

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 channel assigned 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.

Register Clients

To register your {redhat} 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 ›].**

WARNING	To register and use {rhel} 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 ›]**.

{rocky} Client Registration

You can register {rocky} clients to your Uyuni Server. The method and details vary depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see **[Client-configuration › Activation-keys ›]**.

Registering {rocky} Clients

This section contains information about registering Salt clients running {rocky} operating systems.

NOTE	Traditional clients are not available on {rocky} 8. {rocky} 8 clients are only supported as Salt clients.

NOTE	Registering {rocky} clients to Uyuni is tested with the default SELinux configuration of `enforcing` with a `targeted` policy. You do not need to disable SELinux to register {rocky} clients to Uyuni.

Add Software Channels

Before you can register {rocky} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 36. CentOS Channels - CLI

OS Version	Base Channel	Client Channel	AppStream Channel
{rocky} 8	rockylinux8	rockylinux8-uyuni-client	rockylinux8-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. Ensure you specify the correct architecture:

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

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

NOTE   The client tools channel provided by `spacewalk-common-channels` is sourced from {uyuni} and not from {suse}.

IMPORTANT   For {rocky} 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

NOTE   You might notice some disparity in the number of packages available in the AppStream channel between upstream and the Uyuni channel. You might also see different numbers if you compare the same channel added at a different point in time. This is due to the way that {rocky} manages their repositories. {rocky} removes older version of packages when a new version is released, while Uyuni keeps all of them, regardless of age.

If you are using modular channels, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the `spacecmd` package will fail.

IMPORTANT   The AppStream repository provides modular packages. This results in the Uyuni {webui} showing incorrect package information. You cannot perform package operations such as installing or upgrading directly from modular repositories using the {webui} or API.

You can use the AppStream filter with content lifecycle management (CLM) to transform modular repositories into regular repositories. Make sure to include `python:3.6` using an AppStream filter if you want to use `spacecmd` on the clients.

Alternatively, you can use Salt states to manage modular packages on Salt clients, or use the `dnf` command on the client. For more information about CLM, see **[Administration › Content-lifecycle ›]**.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

Create an Activation Key

You need to create an activation key that is associated with your {rocky} channels.

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

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

> **NOTE**
>
> If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

Register Clients

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

Manage Errata

When you update {rocky} clients, the packages include metadata about the updates.

Ubuntu Client Registration

You can register {ubuntu} clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

Before you start, ensure that the client has the date and time synchronized correctly with the Uyuni Server.

You must also have created an activation key. For more information about creating activation keys, see **[Client-configuration › Activation-keys ›]**.

Registering {ubuntu} 20.04 Clients

This section contains information about registering Salt clients running {ubuntu} 20.04 LTS operating systems.

IMPORTANT   Canonical does not endorse or support Uyuni.

NOTE   {ubuntu} is supported for Salt clients only. Traditional clients are not supported.

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

Add Software Channels

Before you register {ubuntu} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 37. Ubuntu Channels - CLI

OS Version	Base Channel	Main Channel	Updates Channel	Security 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-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_label>
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. Synchronize the channels:

```
spacewalk-repo-sync -p <base_channel_label>
```

3. Ensure the synchronization is complete before continuing.

> **IMPORTANT** You need all the new channels fully synchronized before bootstrapping any Ubuntu client.

Check Synchronization Status

Procedure: Checking Synchronization Progress

1. In the Uyuni {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

> **NOTE** {ubuntu} channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

> **NOTE** If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

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 is bootstrapping the client in the {webui}:

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

NOTE	This procedure grants root access without requiring a password, which is required for registering the client. When the client is successfully installed it runs 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.

IMPORTANT	Canonical does not endorse or support Uyuni.

NOTE	{ubuntu} is supported for Salt clients only. Traditional clients are not supported.

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

Add Software Channels

Before you register {ubuntu} clients to your Uyuni Server, you need to add the required software channels, and synchronize them.

The channels you need for this procedure are:

Table 38. Ubuntu Channels - CLI

OS Version	Base Channel	Main Channel	Updates Channel	Security Channel	Universe Channel	Universe Updates Channel	Universe Security Updates Channel	Client Channel
{ubuntu} 16.04	ubuntu-1604-pool-amd64-uyuni	ubuntu-1604-amd64-main-uyuni	ubuntu-1604-amd64-updates-uyuni	ubuntu-1604-amd64-security-uyuni	ubuntu-1604-amd64-universe-uyuni	ubuntu-1604-amd64-universe-updates-uyuni	ubuntu-1604-amd64-universe-security-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-universe-security-uyuni	ubuntu-1804-amd64-uyuni-client

NOTE

{ubuntu} 16.04 is now at end-of-life, and the ISO images provided in the repository are out of date. Bootstrapping new {ubuntu} 16.04 clients using these packages will fail. If you need to bootstrap new {ubuntu} 16.04 clients, follow the troubleshooting procedure in **[Client-configuration › Tshoot-clients ›]**.

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>
```

2. Synchronize the channels:

```
mgr-sync refresh --refresh-channels
```

IMPORTANT

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 {webui}, navigate to menu: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 need to check all the base and child channel log files to be sure that the synchronization is complete.

> **NOTE** {ubuntu} channels can be very large. Synchronization can sometimes take several hours.

Trust GPG Keys on Clients

By default, operating systems trust only their own GPG keys when they are installed, and do not trust keys provided by third party packages. The clients can be successfully bootstrapped without the GPG key being trusted. However, you cannot install new client tool packages or update them until the keys are trusted.

Salt clients are set to trust {suse} tools channels GPG keys when they are bootstrapped. For all other clients and channels, you need to manually trust third party GPG keys.

> **NOTE** If you are bootstrapping Salt clients from the Uyuni {webui}, you can use a Salt state to trust the key. For more information on custom Salt states, see **[Salt › Custom-states ›]**.

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 channel assigned 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.

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 is bootstrapping the client in the {webui}:

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

> **NOTE**
> This procedure grants root access without requiring a password, which is required for registering the client. When the client is successfully installed it runs 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 ›]**.

Register Clients to a Proxy

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

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

Within the {webui}, proxy pages 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 menu: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

menu: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 need to repeat the registration process from the beginning.

Register Clients to a Proxy with the {webui}

You can register Salt clients to the Uyuni Proxy using the {webui}.

NOTE	A bootstrap repository is needed for non-SLE clients in general and for SLE clients before version 15. A bootstrap repository 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 with the {webui}

1. In the Uyuni {webui}, navigate to menu: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 `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 is configured to use SSH for its connection to the server, and no other connection method is configured.

10. Click btn:[Bootstrap] to begin registration.

When the bootstrap process has completed, your client is listed at menu:Systems[System List].

Register on the Command Line (Salt)

Instead of the {webui}, 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.

> **NOTE**
>
> 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)

You can register Salt or traditional clients through the Uyuni Proxy with a bootstrap script. This 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 Uyuni 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. These parameters 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 {webui}. For more information, see [**Client-configuration › Activation-keys ›]**.

2. On the proxy, execute the `mgr-bootstrap` command line tool as {rootuser}. 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 directly on the clients or from the proxy with `ssh`. Replace `<bootstrap>` with the name of the bootstrap script and `<client.example.com>` with the host name of your client:

   ```
   cat <bootstrap> | ssh root@<client.example.com> /bin/bash
   ```

Registering clients on a public cloud

When you have your Uyuni Server set up, you are ready to start registering clients.

Prepare on-demand images

An instance started from an on-demand image provided by {suse} is automatically registered, and the update infrastructure and {sle} modules are activated. To use your on-demand image as a Uyuni client, you need to disable this automation before you begin.

Procedure: Preparing on-demand images

1. Log in to the on-demand instance.

2. At the command prompt, as root, remove the registration data and repositories:

   ```
   registercloudguest --clean
   ```

3. Remove the `cloud-regionsrv-client` package:

   ```
   zypper rm cloud-regionsrv-client
   ```

4. Remove additional packages specific to your cloud provider:

   ◦ Amazon EC2:

```
zypper rm regionServiceClientConfigEC2 regionServiceCertsEC2
```

- ◦ Google Compute Engine:

```
zypper rm cloud-regionsrv-client-plugin-gce regionServiceClientConfigGCE
regionServiceCertsGCE
```

- ◦ Microsoft Azure:

```
zypper rm regionServiceClientConfigAzure regionServiceCertsAzure
```

For instructions on registering Uyuni to {scc}, see **[Installation › Server-setup ›]**.

Register clients

In the Uyuni {webui}, navigate to menu:Systems[Bootstrapping], then fill in the `Host`, `SSH Port`, `User`, and `Password` fields. Make sure you use stable FQDNs for the `Host` field, or Uyuni cannot find your host when your Public Cloud gives you a different short-lived FQDNS.

> **NOTE**
>
> If you are attempting to bootstrap traditional clients, check that you can resolve the host name of the server while you are logged in to the client. You might need to add the FQDN of the server to `/etc/hosts` local resolution file on the client. Check using the `hostname -f` command with the local IP address of the server.

Public cloud images usually do not allow SSH login with username and password, but only SSH with a certificate. If you want to use bootstrap from the {webui}, you need to enable SSH login with username and SSH key. You can do this by navigating to menu:Systems[Bootstrapping] and changing the authentication method.

If your cloud provider is Microsoft Azure, you can log in with username and password. To do this, you need to allow the AzureUser to run commands as root without a password. To do this, open the `/etc/sudoers.d/waagent` file, and add or edit this line:

```
AzureUser ALL=(ALL) NOPASSWD: ALL
```

> **WARNING**
>
> Allowing the AzureUser to run commands as root without a password carries a security risk. Use this method for testing only. Do not do this for production systems.

When the bootstrap process has completed successfully, your client is listed at menu:Systems[System List].

- If you want more control over the process, have to register many clients, or are registering traditional clients, create a bootstrap script. For more information, see **[Client-configuration › Registration-bootstrap ›]**.

- For Salt clients and even more control over the process, executing single commands on the command line can be useful. For more information, see **[Client-configuration › Registration-cli ›]**.

- When registering clients launched from a public cloud image (for example, AWS AMI), you need to do some additional configuration to prevent them from over-writing each other. For more information about registering clones, see **[Administration › Tshoot-registerclones ›]**.

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.

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

Delete Clients

If you need to remove a client from your Uyuni Server, you can use the {webui} to delete it. This procedure works for both traditional and Salt clients.

Procedure: Deleting a Client

1. In the Uyuni {webui}, navigate to menu:Systems[System List] and select the client to delete.

2. Click btn:[Delete System].

3. Check the details and click btn:[Delete Profile] to confirm.

4. For Salt clients, Uyuni attempts to clean up additional configuration. If the client cannot be contacted, you are given the option to cancel the deletion, or delete the client without cleaning up the configuration files.

You can also delete multiple clients using the system set manager. For more information about the system set manager, see **[Client-configuration › System-set-manager ›]**.

Operating System Installation

Generally, you register clients that are already running. You might have installed these machines manually just before registering them to Uyuni, or they might be pre-existing systems that were installed before you added Uyuni to your environment.

Alternatively, you can use Uyuni to help you install an operating system and register it to Uyuni in one go. This method is partially or totally automated, so you can save time answering installer questions, and is especially useful if you have many clients you want to install and register.

There are several ways to install an operating system from Uyuni:

- in-place, on clients that are already registered;

- over the network, using PXE boot;

- preparing an installation CD-ROM or an USB key, and then going to the machine to boot on that medium;

- as part of the Uyuni {smr} solution.

The in-place reinstallation method assumes that a previous operating system has already been installed on the client, and that the client has already been registered to Uyuni.

For information about the in-place installation method, see **[Client-configuration › Autoinst-reinstall › Reinstall Registered Systems]**.

The network boot installation method works on unformatted machines. However, it can only be performed in certain network configurations:

- the Uyuni Server, or one of its proxies, are on the same local network as the machine you want to install, or you have a DHCP relay that allows you to cross all routers in between;

- you are able to set up a new DHCP server or to configure an existing one;

- the client to install is able to boot with PXE, and you can configure it to do so.

For information about the network boot method, see **[Client-configuration › Autoinst-pxeboot › Install via the Network]**.

The removable medium method allows you to bypass these network constraints. However, it assumes the machine is able to read CD-ROMs or USB keys, and boot from them. It also requires physical access to the client machine.

For information about the removable media method, see **[Client-configuration › Autoinst-cdrom › Install via a CD-ROM or a USB key]**.

For information about the Uyuni {smr} approach, see **[Retail › Retail-overview › Retail Guide]**.

> **NOTE**
> Autoinstallation of {ubuntu} and {debian} clients is not supported. These operating systems must be installed manually.

> **NOTE**
> The autoinstallation features of SUSE Manager are based on a software named Cobbler. For more information, see https://cobbler.github.io/quickstart/.

Reinstall Registered Systems
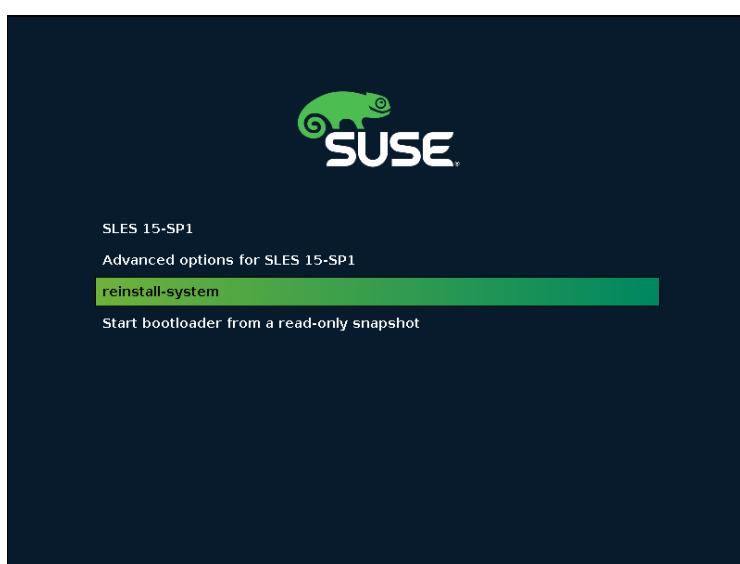
The in-place reinstallation starts from the local client system. There is therefore no need for the client to be able to boot over network with PXE.

To reinstall a registered client in-place, you must define an autoinstallable distribution and an autoinstallation profile. For information, see **[Client-configuration › Autoinst-distributions › Autoinstallable Distributions]** and **[Client-configuration › Autoinst-profiles › Autoinstallation Profiles]**.

When you have defined the autoinstallation profile and distribution, you can perform the reinstallation.

Procedure: Reinstall an Already Registered Client

1. In the Uyuni {webui}, navigate to menu:Systems[Systems List], select the client to reinstall, and go to the menu:Provisioning[Autoinstallation > Schedule] subtab.

2. Select the autoinstallation profile that you prepared, select a proxy if needed, and click btn:[Schedule Autoinstallation and Finish].

3. If your client is a traditional client, and you have not configured osad, you need to wait until the job is fetched.

4. You can monitor progress of the installation by navigating to menu:Provisioning[Autoinstallation > Session Status], or on the client directly. The client reboots, and in the boot menu selects a new choice called `reinstall-system`.
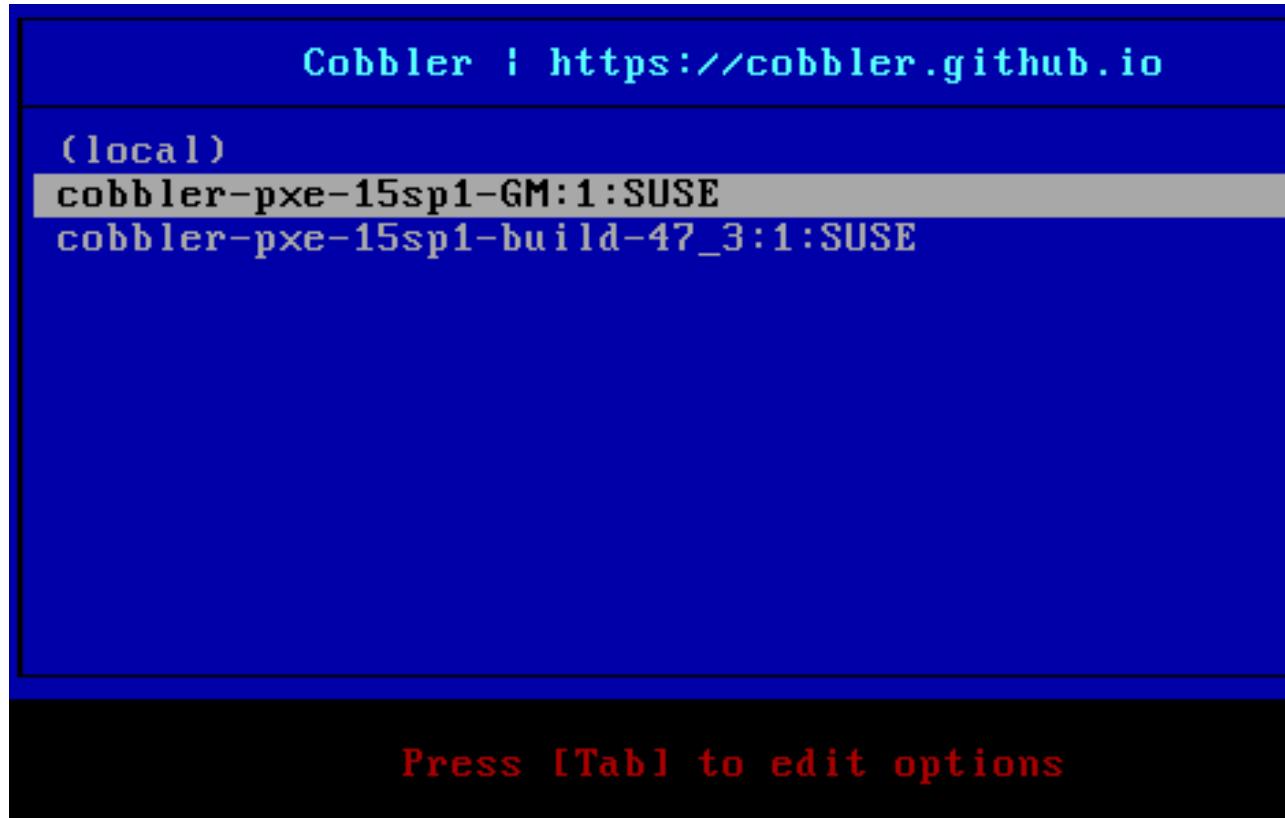
The installation then proceeds over HTTP protocol.

Install via the Network

During a network boot installation:

1. The client boots in PXE mode.

2. A DHCP server provides the client with an IP address and mask, the address of an installation server, and the name of a bootloader file on that server.

3. The client downloads the bootloader file via the TFTP protocol from the installation server and executes it.

4. The client is presented with a choice of possible profiles to install from a menu, or starts automatically installing one of the profiles.

5. The client downloads via the TFTP protocol a kernel and an initial RAM disk for the distribution matching that profile.

6. The installation kernel starts an installation program, either {kickstart} or {ay}. From now on it uses resources provided on the server via the HTTP protocol.

7. The distribution is installed automatically according to the {kickstart} or {ay} profile.

8. The profile calls a code snippet that registers the client to the Uyuni Server, either as a traditional or Salt client.

The installation server can either be the Uyuni Server or one of its proxies. To install from a proxy, you must synchronize the TFTP tree between the server and the proxy before you begin.

The DHCP server may also provide other configuration information to the client, like a host name, the

address of a router and the address of a domain name server. Some of this information might be needed for the autoinstallation, for example if you specify the installation server by its domain name.

In the PXE boot menu, the first choice is `Local boot`. If you select this, the boot process continues from the local disk drives. This option is selected automatically if no profile was selected after a certain time. This is a security measure to prevent from starting an autoinstallation if no human operator is there to select one of the profiles.

Alternatively, the installation can start automatically from one of the profiles without manual intervention. This is called "unattended provisioning".

The "bare metal" functionality is a kind of unattended provisioning based on PXE boot. In this scenario, the bootloader file only registers the client at the Uyuni Server and does not start the installation. You can then trigger an in-place reinstallation later.

Procedure: Installing with PXE Boot

1. Prepare the DHCP server, see Prepare the DHCP Server.

2. Prepare an autoinstallable distribution, see **[Client-configuration › Autoinst-distributions › Autoinstallable Distributions]**.

3. Prepare an autoinstallation profile, see **[Client-configuration › Autoinst-profiles › Autoinstallation Profiles]**.

4. Reboot the client, and chose a profile to install.

Some other steps are optional. To use a proxy as the installation server, see Synchronize the TFTP Tree with Proxies. For unattended provisioning, see **[Client-configuration › Autoinst-unattended › Unattended Provisioning]**.

Prepare the DHCP Server

The PXE boot process uses DHCP to find the TFTP server. The Uyuni Server or its proxies can act as such a TFTP server.

You must have administrative access to the network's DHCP server. Edit the DHCP configuration file so that it points to the installation server as the TFTP boot server.

Example: Configuring the ISC DHCP Server

1. On the DHCP server, as root, open the `/etc/dhcpd.conf` file.

2. Modify the declaration for your client:

```
host myclient { (...)
             next-server 192.168.2.1;
             filename "pxelinux.0"; }
```

1. Save the file and restart the `dhcpd` service.

This example directs the PXE client `myclient` to the installation server at `192.168.2.1` and instructs it to retrieve the `pxelinux.0` bootloader file.

Alternatively, if your DHCP server is registered at Uyuni, you can configure it using the DHCPd formula instead:

Example: Configuring the ISC DHCP Server Using the DHCPd Formula

1. Navigate to menu:Systems[System List], select the client you want to change, and go to the `Formulas` tab to enable the DHCPd formula.

2. Go to the formula's `Dhcpd` tab, and in the `Next Server` field, enter the hostname or IP address of the installation server.

3. In the `Filename EFI` field, type `grub/grub.efi` to enable EFI PXE support.

4. In the `Filename` field, type `pxelinux.0` to enable legacy BIOS support.

5. Click btn:[Save Formula] to save your configuration.

6. Apply the highstate.

This sets up a global PXE server for all the hosts, you can also have per-host settings. For more information about the DHCPd formula, see **[Salt › Formula-dhcpd › DHCPd Formula]**.

Synchronize the TFTP Tree with Proxies

You can synchronize the TFTP tree on the Uyuni Server with the Uyuni Proxy. For synchronization, HTTPS port 443 must be open.

> **WARNING**   Every added proxy slows down the tree synchronization.

Procedure: Synchronizing TFTP between Server and Proxy

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

```
zypper install susemanager-tftpsync
```

1. On the Uyuni Proxy, at the command prompt, as root, install the `susemanager-tftpsync-recv` package:

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

1. On the proxy, as root, run the `configure-tftpsync.sh` script. The script interactively asks 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.

2. On the server, as root, run the `configure-tftpsync.sh` script.

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

1. Run the `cobbler sync` command on the server to push the files to the proxy. This fails if you have not configured the proxies correctly.

If you want to change the list of proxies later on, you can use the `configure-tftpsync.sh` script to edit them.

> **NOTE** 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`.

Install via a CD-ROM or a USB Key

For clients that are not yet registered to Uyuni, and if network boot over PXE is not an option, a bootable CD-ROM or USB key can be used to install the system.

One option to prepare such a removable medium is to use Cobbler. For information about using Cobbler to prepare an ISO image, see Build an ISO Image With Cobbler.

The other option is to use mechanisms specific to the distribution:

- For {suse} systems, prepare an ISO image using KIWI. For information, see Build a SUSE ISO Image With KIWI.

- For {redhat} systems, use `mkisofs`. For information, see Build a RedHat ISO Image With mkisofs.

In all cases, you use the resulting image to burn a CD-ROM or prepare a USB key.

Build an ISO Image With Cobbler

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

> **NOTE** Building ISOs with Cobbler is not supported on {ibmz}.

In order to prepare an ISO image with Cobbler, you need to prepare a distribution and a profile, similar to using network boot over PXE. For information about creating a distribution, see **[Client-configuration › Autoinst-distributions › Autoinstallable Distributions]**. For information about creating a profile, see **[Client-configuration › Autoinst-profiles › Autoinstallation Profiles]**.

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"
```

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

Build a SUSE ISO Image With KIWI

KIWI is an image creation system. You can use KIWI to create a bootable ISO image to be used by the target system for installation of a {suse} system. When the system is rebooted or switched on, it boots from the image, loads the {ay} configuration from your Uyuni, and installs {sles} according to the {ay} profile.

To use the ISO image, boot the system and type `autoyast` at the prompt (assuming you left the label for the {ay} boot as `autoyast`). Press kbd:[Enter] to begin the {ay} installation.

For more information about KIWI, see http://doc.opensuse.org/projects/kiwi/doc/.

Build a RedHat ISO Image With mkisofs

You can use `mkisofs` to create a bootable ISO image to be used by the target system for installation of a {redhat} system. When the system is rebooted or switched on, it boots from the image, loads the {kickstart} configuration from your Uyuni, and installs {rhel} according to the {kickstart} profile.

Procedure: Building a Bootable ISO With mkisofs

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 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, to prevent errors occurring.

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

```
ksdevice=eth0
```

It is possible to change the distribution for a Kickstart profile within a family, such as {rhel} AS 4 to {rhel} 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. Alternatively prepare an USB key and insert it.

7. Boot the system and type `ks` at the prompt (if you left the label for the Kickstart boot as 'ks').

8. Press kbd:[Enter] to start {kickstart}.

Autoinstallable Distributions

The autoinstallation process relies on an several files to initiate the installation. These files include the Linux kernel, an initial RAM disk, and other files required to boot the operating system in installation mode.

You can extract the needed files from a DVD image. For information, see Distribution Based on an ISO Image.

Alternatively, you can install `tftpboot-installation` package. For information, see Distribution Based on a RPM Package.

You must also have a base channel synchronized on your Uyuni Server for the same operating system version as those files.

When you have the files ready, and the base channel synchronized, you need to declare the distribution. This operation associates the installation files to the base channel. The distribution can be referred to by one or more installation profiles. For information, see Declare an Autoinstallable Distribution.

Distribution Based on an ISO Image

This method assumes you have installation media for the operating system you want to install on the clients. This is usually a DVD `.iso` image that contains the Linux kernel, an `initrd` file, and other files required to boot the operating system in installation mode.

Procedure: Importing Files from Installation Media

1. Copy the installation media to your Uyuni Server. For {suse} operating systems, you can download installation media from https://www.suse.com/download/.

2. Loop-mount the ISO image, and copy its contents somewhere:

```
# mount -o loop,ro <image_name>.iso /mnt
# mkdir -p /srv/www/distributions
# cp -a /mnt /srv/www/distributions/<image_name>
# umount /mnt
```

```
Take a note of the file path.
You will need it when you declare the distribution to {productname}.
```

Distribution Based on a RPM Package

This method works on {suse} systems. It is simpler than importing contents from an installation media, because it uses prepackaged files for your installation system.

Procedure: Extracting Files from an Installation Package

1. On the Uyuni Server, install the package whose name starts with `tftpboot-installation`. You can determine its exact name with the command `zypper se tftpboot-installation`

2. Determine where the installation files went with the command `ls -d /usr/share/tftpboot-installation/*`. Take a note of the file path. You will need it when you declare the distribution to Uyuni.

This procedure prepares the installation of the same operating system version as the one that powers your Uyuni Server. If you want to install a different operating system or version on the client, you need to manually get the package `tftpboot-installation-*` from the distribution it belongs to. In the `Package Search` input box Uyuni, search the packages whose names start with `tftpboot-installation`, then look at the package's details. They show the local path below `/var/spacewalk/`.

Declare an Autoinstallable Distribution

The next step after extracting the autoinstallation files is to declare an autoinstallable distribution.

Procedure: Declaring an Autoinstallable Distribution

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

2. Click `Create Distribution`, and complete these fields:

   ◦ In the `Distribution Label` field, enter a name to identify your autoinstallable distribution.

   ◦ In the `Tree Path` field, enter the path to the installation media saved on your Uyuni Server.

   ◦ Select the matching `Base Channel`. This must match the installation media.

   ◦ Select the `Installer Generation`. This must match the installation media.

   ◦ OPTIONAL: Specify kernel options to use when booting this distribution. There are multiple ways to provide kernel options. Only add options here that are generic for the distribution.

3. Click btn:[Create Autoinstallable Distribution].

The installation files that you prepared might not contain the packages you need to install. If they are not included, add `useonlinerepo=1` to the `Kernel Options` field.

The package repositories contain metadata that can be unsigned. If the metadata is unsigned, add `insecure=1` to the `Kernel Options` field, or use your own GPG key as explained in **[Client-configuration › Autoinst-owngpgkey › Use Your Own GPG Key]**.

These kernel options are needed for example when you use the "online installer" ISO images instead of the full DVD, or when you use the `tpboot-installation` package.

Navigate to menu:Systems[Autoinstallation > Distributions] to manage your autoinstallable distributions.

NOTE	You can autoinstall Uyuni Proxies in the same way as {sle} clients. Make sure you use the {sle} installation media, and choose the `SLE-Product-SUSE-Manager-Proxy-4.2-Pool for x86_64` base channel.

Autoinstallation Profiles

An autoinstallation profile determines how the operating system will be installed. For example, you can specify additional kernel parameters to be passed to the installer.

The most important part of the profile is the "autoinstallation file". When you perform an installation manually, you must provide information to the installer, such as partitioning and networking information and user details. The autoinstallation file is a method of providing this information in a scripted form. This type of file is sometimes also referred to as an "answers file".

Within Uyuni, you can use two different types of profiles, depending on the operating system of the clients you want to install:

• For {sle} clients, use {ay}.

• For {rhel} clients, use {kickstart}.

You can use both {ay} and {kickstart} profiles if you want to install clients with different operating systems.

For information about how to declare profiles, see Declare the Profile

For information about {ay} profiles, see AutoYast Profiles.

For information about {kickstart} profiles, see Kickstart Profiles.

The autoinstallation file contained in the profile can include variables and code snippets. For information about variables and code snippets, see Templates Syntax.

Declare the Profile

When you have prepared an autoinstallation file and distribution, you can create profiles to manage autoinstallation on your Uyuni Server. For a given distribution, the profile will determine how to install this distribution.

One way to create a profile is to upload an {ay} or {kickstart} file. Alternatively, for {kickstart} only, you can use the {webui} wizard.

Procedure: Creating an Autoinstallation Profile by Upload

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

2. Click btn:[Upload Kickstart/Autoyast File].

3. In the `Label` field, type a name for the profile. Do not use spaces.

4. In the `Autoinstall Tree` field, select the autoinstallable distribution to use for this profile.

5. In the `Virtualization Type` field, select the type of virtualization to use for this profile, or select `None` if you do not want to use this profile to create a new virtual machine.

6. Copy the contents of your autoinstallation file into the `File Contents` field, or upload the file directly using the `File to Upload` field. For more information about the details to include here, see AutoYast Profiles or Kickstart Profiles.

7. Click btn:[Create] to create the profile.

Procedure: Creating a {kickstart} Profile by Wizard

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

2. Click btn:[Create Kickstart Profile].

3. In the `Label` field, type a name for the profile. Do not use spaces.

4. In the `Base Channel` field, select the base channel to use for this profile. This field is populated from the distributions available. If the base channel you need is not available, check that you have created the distribution correctly.

5. In the `Virtualization Type` field, select the type of virtualization to use for this profile, or select `None` for no virtualization.

6. Click btn:[Next].

7. In the `Distribution File Location` type the path to the installation media installed on the Uyuni Server.

8. Click btn:[Next].

9. Provide a password for the root user on the client.

10. Click btn:[Finish].

11. Review the details of your new profile, and customize as required.

When you are creating your autoinstallation profile, you can check `Always use the newest Tree for this base channel`. This setting allows Uyuni to automatically pick the latest distribution that is associated with the specified base channel. If you add new distributions later, Uyuni uses the most recently created or modified.

Changing the Virtualization Type usually requires changes to the profile bootloader and partition options. This can overwrite your customization. Verify new or changed settings before saving them, by navigating to the `Partitioning` tab.

The kernel options from the distribution and the profile are combined.

You can change the details and settings of your autoinstallation profiles by navigating to menu:Systems[Autoinstallation > Profiles] and clicking the name of the profile you want to edit. Alternatively, navigate to menu:Systems[System List], select the client you want to provision, and navigate to the menu:Provisioning[Autoinstallation] subtab.

AutoYast Profiles

An {ay} profile consists of a `Label` that identifies the profile, an `Autoinstall Tree` that points to an autoinstallable distribution, various options, and, most importantly, an {ay} installation file.

The {ay} installation file is an XML file that give directions to the {ay} installer. {ay} calls it a "control file". For the full syntax of {ay} installation files, see https://doc.opensuse.org/projects/autoyast/#cha-configuration-installation-options.

{suse} provides templates of {ay} installation files that you can use as a starting point for your own custom files. You will find them at https://github.com/SUSE/manager-build-profiles in the directory named `AutoYast`. Each of these profiles requires you to set some variables before you use it. Check the `README` file included with the script to determine which variables you need. For more information about using variables in {ay} scripts, see Variables.

Here are the most important sections in the {ay} installation file for installing with SUSE Manager: * `<add-on>` section allows to add child channels to the installation * `<general>$SNIPPET('spacewalk/sles_no_signature_checks')</general>`` disables signature checks * <software> `allows to specify the correct product for the Unified Installer *` <init-scripts config:type="list">$SNIPPET('spacewalk/minion_script')</init-scripts>`` allows the client to register to Uyuni as a Salt minion.

For more information about {ay}, see https://doc.opensuse.org/projects/autoyast/.

A more recent, Salt-based alternative to AutoYast, is Yomi. For information about Yomi, see **[Salt › Yomi › Install with Yomi]**.

Kickstart Profiles

{kickstart} profiles offer a large number of configuration options. To create these profiles, you can upload them, or use a dedicated wizard.

{kickstart} profiles allow you to use file preservation lists. If you have many custom configuration files located on a client you want to reinstall with {kickstart}, you can save them as a list, and associate that list with the {kickstart} profile.

Procedure: Creating a File Preservation List

1. In the Uyuni {webui}, navigate to menu:Systems[Autoinstallation > File Preservation] and click btn:[Create File Preservation List].

2. Enter a suitable label, and list absolute paths to all files and directories you want to save.

3. Click btn:[Create List].

4. Include the file preservation list in your {kickstart} profile. Navigate to menu:Systems[Autoinstallation > Profiles] and select the profile you want to edit, go to the menu:System Details[File Preservation] subtab, and select the file preservation list to include.

NOTE File preservation lists are limited to a total size of 1 MB. Special devices like `/dev/hda1` and `/dev/sda1` cannot be preserved. Only use file and directory names, you cannot use regular expression wildcards.

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

Templates Syntax

Parts of your installation file are replaced during the installation. Variables are replaced with single values, and code snippets are replaced with whole sections of text. Escaped symbols or sections are not replaced.

A template engine called Cheetah allows Cobbler to do these replacements. This mechanism allows you to reinstall large numbers of systems, without having to manually create profiles for each of them.

You can create autoinstallation variables and code snippets within the Uyuni {webui}. Within a profile, the `Autoinstallation File` tab allows you to see the result of the substitutions.

For information about variables, see Variables. For information about code snippets, see Code Snippets. For information about escaping symbols or whole sections, see Escaping.

Variables

Autoinstallation variables can be used to substitute values into {kickstart} and {ay} profiles. To define a variable, from the profile, navigate to the `Variables` subtab, and create a `name=value` pair in the text box.

For example, you could create a variable that holds the IP address of the client, and another that holds the address of its gateway. Those variables can then be defined for all the clients installed from the same profile. To do that, add these lines to the `Variables` text box:

```
ipaddr=192.168.0.28
gateway=192.168.0.1
```

To use the variable, prepend a `$` sign in the profile to substitute the value. For example, the `network` part of a {kickstart} file may look like the following:

```
network --bootproto=static --device=eth0 --onboot=on --ip=$ipaddr \
  --gateway=$gateway
```

The `$ipaddr` is resolved to `192.168.0.28`, and the `$gateway` to `192.168.0.1`.

In installation files, variables use a hierarchy. System variables take precedence over profile variables, which in turn take precedence over distribution variables.

Code Snippets

Uyuni comes with a large number of predefined code snippets. Navigate to menu:Systems[Autoinstallation > Autoinstallation Snippets] to see the list of existing snippets.

Use a snippet by inserting the `$SNIPPET()` macro in your autoinstallation file. For example, in {kickstart}:

```
$SNIPPET('spacewalk/rhel_register_script')
```

Or, in {ay}:

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

The macro is parsed by Cobbler and substituted with the contents of the snippet. You can also store your own code snippets to use in autoinstallation files later on. Click btn:[Create Snippet] to create a new code snippet.

This example sets up a {kickstart} 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 vgname=vg00 --fstype ext3 --size=5000
```

Use the snippet with, for example:

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

Escaping

If the autoinstallation file contains shell script variables like `$(example)`, the content needs to be escaped with a backslash: `\$(example)`. Escaping the `$` symbol prevents 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` 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)
```

Unattended Provisioning

The "bare metal" functionality allows you to register any new machine as soon as it connects to the local network, using a generic PXE boot image. You then go to Uyuni {webui} and assign a profile to this machine. The next time the client boots, the operating system is installed according to that profile. For information about the bare metal provisioning, see Bare Metal Provisioning.

If you don't want to use the bare metal functionality, you can still declare manually the system at Uyuni. The Uyuni API allows you to create system records for systems as if they were collected by the bare metal functionality. For information about declaring systems with the API, see Create a System Record Manually.

Bare Metal Provisioning

When you have the bare metal provisioning option enabled, any client connected to the Uyuni network is automatically added to the organization as soon as it is powered on. When this operation is complete, the client is shut down, and it appears in the `Systems` list, ready to be installed.

Procedure: Enable the Bare Metal Functionality

1. In the Uyuni {webui}, navigate to menu:Admin[Manager Configuration > Bare-metal systems].

2. Click btn:[Enable adding to this organization].

New clients that are powered on are added to the organization that belongs to the administrator who enabled the bare metal feature. They are of "bootstrap" type, and still need to be provisioned to become regular clients.

To change the organization new 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 migrate already registered systems to other organizations, using the `Migrate` tab.

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

Procedure: Provision "Bootstrap" Type clients

1. In the Uyuni {webui}, navigate to `Systems`, select the client to provision, and go to the menu:Provisioning[Autoinstallation] tab.

2. Select the {ay} profile to use, and click btn:[Create PXE installation configuration]. This option creates a system entry at Cobbler.

3. Power on the client.

The server uses TFTP to provision the new client, so the appropriate port and networks must be configured correctly in order for provisioning to be successful.

Create a System Record Manually

You can use an API call to declare an association between a client identified by its MAC address and an autoinstallation profile. Next time the system reboots, it starts the installation based on the specified profile.

Procedure: Reinstallation From a Manually Declared Profile

1. On the Uyuni Server, at the command prompt, use the `system.createSystemRecord` API call. In this example, replace `name` with the name of your client, `<profile>` with the profile label, `<iface>` with the name of the interface on the client such as `eth0`, and `<hw_addr>` with its hardware address such as `00:25:22:71:e7:c6`:

```
$ spacecmd api -- --args '["<name>", "<profile>", "", "", \
  [ {"name": "<iface>", "mac": "<hw_addr>"} ]]' \
  system.createSystemRecord
```

2. Power on the client. It boots from the network, and the correct profile is selected for installation.

This command creates a system record at Cobbler. You may also specify additional parameters, like kernel options, the IP address of the client, its domain name, etc. For more information, see the API documentation for `createSystemRecord call`.

Use Your Own GPG Key

If the repositories you are using for autoinstallation have unsigned metadata, you usually have to use the `insecure=1` kernel parameter as an option of the autoinstallable distribution, and use a `spacewalk/sles_no_signature_checks` code snippet in the {ay} installation file.

A safer alternative is to provide your own GPG key.

Procedure: Include your own GPG key

1. Create a GPG key.

2. Use it to sign the package's metadata.

3. Add it to the initial RAM disk of your installation media.

For information on how to create the key and sign the metadata with it, see **[Administration › Repo-metadata › Signing Repository Metadata]**.

For information on how to add the key to installation media used for network boot, see Own GPG Key for PXE Boot.

For information on how to add the key to installation media used for a boot from CD-ROM, see Own GPG Key in a CD-ROM.

NOTE	This technique applies only to {suse} clients.

NOTE	Once you signed the metadata with your new GPG key, any already onboarded client will not know about the new key. Ideally, you should sign the metadata before you register any client.
	Should you have already onboarded clients that use those repositories, the simplest fix is to disable GPG key checking on them.

Own GPG Key for PXE Boot

The initial RAM disk (`initrd`) used by PXE boot process normally contains {suse}'s GPG key only. We need to add our own key to this file, so it can be used to check the packages.

Procedure: Add a GPG key to the initial RAM disk

1. Create a directory with a path identical to the one that is used during the boot process to find the GPG key:

   ```
   $ mkdir -p tftproot/usr/lib/rpm/gnupg/keys
   ```

2. Copy your GPG key into this directory with the suffix `.asc`:

   ```
   $ cp /srv/www/htdocs/pub/mgr-gpg-pub.key tftproot/usr/lib/rpm/gnupg/keys/mgr-gpg-pub.asc
   ```

3. Inside the top level directory, package the content and append it to the `initrd` that is part of your installation media files:

   ```
   $ cd tftproot; find . | cpio -o -H newc | xz --check=crc32 -c >> /path/to/initrd
   ```

Own GPG key in a CD-ROM

You can modify an installation image with the `mksusecd` utility. This command is contained in the Development Tools module.

Procedure: Add a GPG key to an Installation ISO image

1. Create a directory with a path identical to the one that is used during the boot process to find the GPG key:

   ```
   $ mkdir -p initrdroot/usr/lib/rpm/gnupg/keys
   ```

2. Copy your GPG key into this directory with the suffix `.asc`:

   ```
   $ cp /srv/www/htdocs/pub/mgr-gpg-pub.key initrdroot/usr/lib/rpm/gnupg/keys/mgr-gpg-
   pub.asc
   ```

3. Amend the existing ISO image with `mksusecd`:

   ```
   $ mksusecd --create <new-image>.iso --initrd initrdroot/ <old-image>.iso
   ```

System Set Manager

The system set manager (SSM) is used to perform actions on more than one client at a time. SSM creates ephemeral sets of clients, making it useful for one-off actions that you need to apply to a number of clients. If you want more permanent sets, consider using system groups instead. For more information about system groups, see **[Client-configuration › System-groups ›]**.

The actions available for use in SSM are listed in this table. The icons in this table indicate:

- {check} this action is available in SSM for this client type

- {cross} this action is not available in SSM for this client type

- {question} this action is under consideration for this client type, and may or may not be supported at a later date.

Table 39. Available SSM Actions

Action	Traditional	Salt
List systems	{check}	{check}
Install patches	{check}	{check}
Schedule patch updates	{check}	{check}
Upgrade packages	{check}	{check}
Install packages	{check}	{check}
Remove packages	{check}	{check}
Verify packages	{check}	{cross}
Create groups	{check}	{check}
Manage groups	{check}	{check}
Channel memberships	{check}	{check}
Channel subscriptions	{check}	{cross}
Deploy/diff channels	{check}	{cross}
Autoinstall clients	{check}	{cross}
Tag for snapshot	{check}	{cross}
Remote commands	{check}	{cross}
Power management	{check}	{cross}
Update system preferences	{check}	{check}

Action	Traditional	Salt
Update hardware profiles	{check}	{check}
Update package profiles	{check}	{check}
Set/remove custom values	{check}	{check}
Reboot clients	{check}	{check}
Migrate clients to another organization	{check}	{check}
Delete clients	{check}	{check}

You can select clients for the SSM in several ways:

- Navigate to menu:Systems[System List] and check the clients you want to work with.

- Navigate to menu:Systems[System Groups], and click btn:[Use in SSM] for the system group you want to work with.

- Navigate to menu:Systems[System Groups], check the group you want to work with, and click btn:[Work with Group].

When you have selected the clients you want to work with, navigate to menu:Systems[System Set Manager], or click the `systems selected` icon in the top menu bar.

> **NOTE**
>
> The details in SSM might differ slightly from the details in other parts of the Uyuni {webui}. In SSM, all available updates are shown. This allows you to upgrade to packages that might not be the latest version.

Change Base Channels in SSM

You can use SSM to change the base channel of more than one client at the same time.

> **IMPORTANT**
>
> Changing the base channel significantly changes the packages and patches available to the affected clients. Use with caution.

Procedure: Using SSM to Change Base Channels for Multiple Clients

1. In the Uyuni {webui}, navigate to menu:Systems[System List], check the clients you want to work with, and navigate to menu:Systems[System Set Manager].

2. Navigate to the `Channels` subtab.

3. Locate the current base channel in the list, and verify that the number shown in the `Systems` column is correct. You can click the number in this column to see more details of the clients you are changing.

4. Select the new base channel in the `Desired base Channel` field, and click btn:[Next].

5. For each child channel, select `No change`, `Subscribe`, or `Unsubscribe`, and click btn:[Next].

6. Check the changes you are making, and choose a time for the action to occur.

7. Click btn:[Confirm] to schedule the changes.

System Groups

You can use system groups to make it easier to manage a large number of clients. Groups can be used to perform bulk actions on clients such as applying updates, configuration channels, salt states, or formulas.

You can organize clients into groups in any way that works for your environment. For example, you could organize clients on which operating system is installed, which physical location they are in, or the type of workload they are handling. Clients can be in any number of groups, so you can define your groups in different ways.

When you have clients organized into groups, you can perform updates on all clients in one or more groups, or on intersections between groups. For example, you can define one group for all Salt clients, and another group for all SLES clients. You can then perform updates on all Salt clients, or use the intersection between the groups, and update all Salt SLES clients.

Create Groups

You need to create some groups before you can use them to organize your clients.

Procedure: Creating a New System Group

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

2. Click btn:[Create Group].

3. Give your new group a name and a description.

4. Click btn:[Create Group] to save your group.

5. Repeat for each group you require.

Add Clients to Groups

You can add individual clients to your groups, or add multiple clients at the same time.

Procedure: Adding A Single Client to a Group

1. In the Uyuni {webui}, navigate to menu:Systems[System List] and click the name of the client to add.

2. Navigate to the menu:Groups[Join] tab.

3. Check the group to join and click btn:[Join Selected Groups].

Procedure: Adding Multiple Clients to a Group

1. In the Uyuni {webui}, navigate to menu:Systems[System Groups] and click the name of the group to add clients to.

2. Navigate to the `Target systems` tab.

3. Check the clients to add and click btn:[Add Systems].

Procedure: Adding Multiple Clients to a Group with SSM

1. In the Uyuni {webui}, navigate to menu:Systems[System List] and check each client to add, this adds the clients to the system set manager.

2. Navigate to menu:Systems[System Set Manager], and go to the `Groups` tab.

3. Locate the group to join and check `Add`.

4. Click btn:[Alter Membership].

5. Click btn:[Confirm] to join the clients to the selected group.

For more information about the system set manager, see **[Client-configuration › System-set-manager ›]**.

You can see which clients are in a group by navigating to menu:Systems[System Groups], clicking the name of the group, and navigating to the `Systems` tab. Alternatively, you can see a graphical representation of your system groups by navigating to menu:Systems[Visualization > Systems Grouping].

Work with Groups

When you have your clients arranged into groups, you can use your groups to manage updates. For Salt clients, you can also apply states and formulas to all clients in a group.

In the Uyuni {webui}, navigate to menu:Systems[System Groups]. The list shows an icon if there are updates available for any of the clients in the group. Click the icon to see more information about the updates available and to apply them to the clients.

You can also work with more than one group at a time. Select the groups you want to work with, and click btn:[Work with union] to select every client in every selected group.

Alternatively, you can work on intersections of groups. Select two or more groups, and click btn:[Work with intersection] to select only those clients that exist in all the selected groups. For example, you might have one group for all Salt clients, and another group for all SLES clients. The intersection of these groups would be all Salt SLES clients.

System Types

Clients are categorized by system type. Every client can have both a base system type, and an add-on system type assigned.

Base system types include `Management`, for traditional clients, and `Salt` for Salt clients.

Add-on system types include `Virtualization Host`, for clients that operate as virtual hosts, and `Container Build Host` for clients that operate as a {caasp} build host.

You can adjust the add-on system type by navigating to menu:Systems[System List > System Types]. Check the clients you want to change the add-on system type for, select the `Add-On System Type`, and click either btn:[Add System Type] or btn:[Remove System Type].

You can also change the base system type from `Management` to `Salt`, by re-registering the client.

Change a Traditional Client to Salt Using the {webui}

The simplest method to change a traditional client to a Salt client is to re-register it with the {webui}.

	Changing the base system type requires that you re-register your client. This
WARNING	deletes any customization or configuration on the client, however event history is preserved. It also requires client downtime.

Procedure: Changing a Traditional Client to Salt Using the {webui}

1. In the Uyuni {webui}, navigate to menu:Systems[System List], identify the client you want to change, and take a note of the hostname.

2. Navigate to menu:Systems[Bootstrapping].

3. In the `Host` field, type the hostname of the client to be re-registered.

4. Complete the other fields as required.

5. Click btn:[Bootstrap] to schedule the bootstrap process.

    When the client has completed registration, it shows in the `Systems List` with the system type `Salt`.

Change a Traditional Client to Salt at the Command Prompt

You can use the command prompt to re-register a traditional client as a Salt client. This requires you to delete the packages used by the traditional client. You can then re-register the client using your preferred registration method for Salt clients.

WARNING	Changing the base system type requires that you re-register your client. This deletes any customization or configuration on the client. It also requires client downtime.

Procedure: Changing a Traditional Client to Salt at the Command Prompt

1. On the client to be changed, at the command prompt, use your package manager to remove these packages:

```
spacewalk-check
spacewalk-client-setup
osad
osa-common
mgr-osad
spacewalksd
mgr-daemon
rhnlib
rhnmd
```

2. Use your preferred registration method to re-register the client as a Salt client.

   When the client has completed registration, it shows in the `Systems List` with the system type `Salt`.

Package Management

Clients use packages to install, uninstall, and upgrade software.

To manage packages on a client, navigate to Systems, click the client to manage, and navigate to the menu:Systems[Software > Packages] subtab. The options available in this section vary depending on the type of client you have selected, and its current channel subscriptions.

> **IMPORTANT** When packages are installed or upgraded, licenses or EULAs are automatically accepted.

Most package management actions can be added to action chains. For more about action chains, see **[Reference › Schedule ›]**.

Verify Packages

You can check that packages you have installed on a client match the current state of the database they were installed from. The metadata of the installed package is compared to information in the database, including the file checksum, file size, permissions, owner, group, and type.

Procedure: Verifying Installed Packages

1. In the Uyuni {webui}, navigate to Systems, click the client the package is installed on, and navigate to the menu:Systems[Software > Packages > Verify] subtab.

2. Select the packages you want to verify and click btn:[Verify Selected Packages].

3. When the verification is complete, navigate to menu:Systems[Events > History] to see the results.

Compare Packages

You can compare the packages installed on a client with a stored profile, or with packages installed on another client. When the comparison is made, you can choose to modify the selected client to match.

To compare packages against a profile, you need to have stored a profile. Profiles are created from the packages on a currently installed client. When the profile has been created, you can use it to install more clients with the same packages installed.

Procedure: Creating a Stored Profile

1. In the Uyuni {webui}, navigate to Systems, click the client to base your profile off, and navigate to the menu:Systems[Software > Packages > Profiles] subtab.

2. Click btn:[Create System Profile].

3. Type a name and description for your profile and click btn:[Create Profile].

Procedure: Comparing Client Packages

1. In the Uyuni {webui}, navigate to Systems, click the client to compare, and navigate to the menu:Systems[Software > Packages > Profiles] subtab. To compare with a stored profile, select the

profile and click btn:[Compare].

2. To compare with another client, select the client name and click btn:[Compare] to see a list of differences between the two clients.

3. Check packages you want to install on the selected client, uncheck packages you want to remove, and click btn:[Sync Packages to].

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 {webui}, navigate to menu:Patches[Manage Patches], click btn:[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 to add the new patch to.

4. Click btn:[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 {webui}, navigate to menu: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 btn:[Clone Patches].

4. Select one or more channels to add the cloned patch 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 {webui}, navigate to menu:Patches[Manage Patches], and click the the advisory name of the patch to see the patch details.

2. Navigate to the menu: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 btn:[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 btn:[Add Packages].

5. Confirm the details of the packages, and click btn:[Confirm] to assign them to the patch.

6. Navigate to the menu: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 {webui}, navigate to menu:Patches[Manage Patches].

2. Click the advisory name of the patch to see the patch details.

3. Make the changes as required, and click btn:[Update Patch].

4. To delete a patch, select the patch in the `Patches Management` page, and click btn:[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 is not installed.

If the client has a more recent version of a package already installed, the update is not installed. If the client has an older version of the package installed, the package is upgraded.

Procedure: Applying All Applicable Patches

1. In the Uyuni {webui}, navigate to menu:Systems[Overview] and select the client you want to update.

2. Navigate to the menu:Software[Patches] tab.

3. Click btn:[Select All] to select all applicable patches.

4. Click btn:[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 {webui}, navigate to menu: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 btn:[Apply Patches]. . Confirm the list of clients to perform the update.

Procedure: Applying Multiple Patches to Multiple Clients

1. In the Uyuni {webui}, navigate to menu:Systems[Overview] and check the clients you want to update to add them to the system set manager.

2. Navigate to menu:Systems[System Set Manager] and naviagte to the `Patches` tab.

3. Select the patches you want to apply to the clients and click btn:[Apply Patches].

4. Schedule a date and time for the update to occur, and click btn:[Confirm].

5. To check the progress of the update, navigate to menu:Schedule[Pending Actions].

IMPORTANT	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 prevents 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 {webui}, and a padlock icon is displayed next to the name of the client in the menu:System[System List].

Procedure: System Locking a Traditional Client

1. In the Uyuni {webui}, navigate to the `System Details` page for the client you want to lock.

2. Under `Lock Status`, click btn:[Lock this system]. The client remains locked until you click btn:[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 {webui} or the API, but the actions fail.

> **NOTE** The locking mechanism is not available for Salt SSH clients.

Procedure: System Locking a Salt Client

1. In the Uyuni {webui}, 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 btn:[Save].

3. Navigate to the menu:Formulas[System Lock] tab, check `Lock system`, and click btn:[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 {webui} notifies you. The client remains locked until you remove the system lock formula.

> **NOTE** The system lock formula is enabled by default if {caasp} 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

NOTE	Package locking can be used on several clients, but different feature sets are available. You must differentiate between: * {sle} and {opensuse} (zypp-based) versus {rhel} or {debian} clients, and * traditional versus Salt clients.

Package Locks on Zypp-based Systems

NOTE	Systems with the Zypper package manager have package locking available on traditional and Salt clients.

Package locks are used to prevent unauthorized installation or upgrades to software packages. When a package has been locked, it shows a padlock icon, indicating that it cannot be installed. Any attempt to install a locked package is reported as an error in the event log.

Locked packages cannot be installed, upgraded, or removed, neither through the Uyuni {webui}, nor directly on the client machine using a package manager. Locked packages also indirectly lock any dependent packages.

Procedure: Using Package Locks

1. Navigate to the menu:Software[Packages > Lock] tab on the managed system to see a list of all available packages.

2. Select the packages to lock, and click btn:[Request Lock]. Pick date and time for the lock to activate. By default, the lock is activated as soon as possible. Note that the lock might not activate immediately.

3. To remove a package lock, select the packages to unlock and click btn:[Request Unlock]. Pick date and time as with activating the lock.

Package Locks on {rhel}- and {debian}-like Systems

NOTE	Some {rhel}- and {debian}-like systems have package locking available on Salt clients.

On {rhel}- and {debian}-like systems, package locks are only used to prevent unauthorized upgrades or removals to software packages. When a package has been locked, it shows a padlock icon, indicating that it cannot be changed. Any attempt to change a locked package is reported as an error in the event log.

Locked packages cannot be upgraded or removed, neither through the Uyuni {webui}, nor directly on the client machine using a package manager. Locked packages also indirectly lock any dependent packages.

Procedure: Using Package Locks

1. On the {rhel} 7 systems, install the `yum-plugin-versionlock` package as `root`. On the {rhel} 8 systems, install the `python3-dnf-plugin-versionlock` package as `root`. On {debian} systems, the `apt` tool has the locking feature included.

2. Navigate to the menu:Software[Packages > Lock] tab on the managed system to see a list of all available packages.

3. Select the packages to lock, and click btn:[Request Lock]. Pick date and time for the lock to activate. By default, the lock is activated 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 btn:[Request Unlock]. Pick date and time as with activating the lock.

Configuration Management

You can use configuration files and channels to manage configuration for your clients, rather than configuring each client manually.

Configuration parameters are scripted and stored in configuration files. You can write configuration files directly using the Uyuni {webui}, or you can upload or link to files that exist in other locations.

Configuration files can be centrally managed, or locally managed. Centrally managed configuration files are provided by global configuration channels and can be applied to any client subscribed to the Uyuni Server. Locally managed configuration files are used to override centrally managed configuration settings. They are especially useful for Uyuni users who do not have configuration administration privileges, but need to make changes to the clients that they manage.

Configuration channels are used to organize configuration files. You can subscribe clients to configuration channels, and deploy configuration files as required.

Configuration files are version-controlled, so you can add configuration settings, test them on your clients, and roll back to a previous revision as required. When you have created your configuration channels, you can also perform comparisons between various configuration files, and between revisions of the same configuration file.

Configuration files can be centrally managed, or locally managed. Centrally managed configuration files are provided by global configuration channels. Locally managed configuration files are created or uploaded to Uyuni directly.

The available configuration management features are different for Salt and traditional clients. This table shows the supported features on different client types:

Table 40. Configuration Management Supported Features

Feature	Salt	Traditional
Global Configuration Channels	{check}	{check}
Deploying Files	{check}	{check}
Comparing Files	{question}	{check}
Locally Managed Files	{cross}	{check}
Sandbox Files	{cross}	{check}
Applying the Highstate	{check}	{cross}
File Import from a Client	{cross}	{check}
Configuration Macros	{cross}	{check}

Prepare Traditional Clients for Configuration Management

Traditional clients require some extra preparation to use configuration management. If you installed your traditional client with {ay} or {kickstart} you probably already have the appropriate packages. For other traditional clients, ensure you have installed the relevant tools child channel for your client operating system. For more information about software channels, see **[Client-configuration › Channels ›]**.

The packages you require are:

- `mgr-cfg`: base libraries and functions needed by all `mgr-cfg-*` packages

- `mgr-cfg-actions`: required to run configuration actions scheduled using Uyuni.

- `mgr-cfg-client`: provides a command line interface to the client features of the configuration management system.

- `mgr-cfg-management`: provides a command line interface to manage Uyuni configuration.

You can install these packages during the bootstrap process by navigating to menu:Systems[Activation Keys], clicking the activation key you intend to use during bootstrap, and checking the `Configuration File Deployment` option. For more information about activation keys, see **[Client-configuration › Activation-keys ›]**.

Create Configuration Channels

To create a new central configuration channel:

Procedure: Creating Central Configuration Channel

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

2. Type a name for the channel.

3. Type a label for the channel. This field must contain only letters, numbers, hyphens (`-`) and underscores (`_`).

4. Type a description for the channel that allows you to distinguish it from other channels.

5. Click btn:[Create Config Channel] to create the new channel.

You can also use a configuration channel to manage Salt states on Salt clients.

Procedure: Creating a Salt State Channel

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

2. Type a name for the channel.

3. Type a label for the channel. This field must contain only letters, numbers, hyphens (`-`) and underscores (`_`).

4. Type a description for the channel that allows you to distinguish it from other channels.

5. Type the `SLS Contents` for the `init.sls` file.

6. Click btn:[Create Config Channel] to create the new channel.

Add Configuration Files, Directories, or Symbolic Links

When you have created a configuration channel you can add a configuration file, directory, or symbolic link:

Procedure: Adding a Configuration File, Directory, or Symbolic Link

1. In the Uyuni {webui}, navigate to menu:Configuration[Channels], and click the name of the configuration channel that you want to add a configuration file to, and navigate to the menu:Add Files[Create File] subtab.

2. In the `File Type` field, choose whether you want to create a text file, directory, or symbolic link.

3. In the `Filename/Path` field, type the absolute path to the location where the file should be deployed.

4. If you are creating a symbolic link, type the target file and path in the `Symbolic Link Target Filename/Path` field.

5. Type the `User name` and `Group name` for the file in the `Ownership` field, and the `File Permissions Mode`.

6. If the client has SELinux enabled, you can configure `SELinux contexts` to enable the required file attributes (for example: user, role, and file type).

7. If the configuration file includes a macro, enter the symbol that marks the beginning and end of the macro.

8. Enter the configuration file contents in the `File Contents` text box, using the script drop-down box to choose the appropriate scripting language.

9. Click btn:[Create Configuration File].

Subscribe Clients to Configuration Channels

You can subscribe individual clients to configuration channels by navigating to menu:Systems[System List], selecting the client you want to subscribe, and navigating to the `Configuration` tab. To subscribe multiple clients to a configuration channel, you can use the system set manager (SSM).

Procedure: Subscribing Multiple Clients to Configuration Channels

1. In the Uyuni {webui}, navigate to menu:Systems[Systems List] and select the clients you want to work with.

2. Navigate to menu:Systems[System Set Manager], and go to the menu:Configuration[Subscribe to Channels] subtab to see the list of available configuration channels.

3. OPTIONAL: Click the number in the `Systems currently subscribed` column to see which

clients are currently subscribed to the configuration channel.

4. Check the configuration channels you want to subscribe to, and click btn:[Continue].

5. Rank the configuration channels using the up and down arrows. Where settings conflicts occur between configuration channels, channels closer to the top of the list take precedence.

6. Determine how the channels are applied to the selected clients. Click btn:[Subscribe With Lowest Priority] to add the new channels at a lower priority than currently subscribed channels. Click btn:[Subscribe with Highest Priority] to add the new channels at a higher priority than currently subscribed channels. Click btn:[Replace Existing Subscriptions] to remove existing channels and replace them with the new channels.

7. Click btn:[Apply Subscriptions].

NOTE | If new configuration channel priorities conflict with existing channels, the duplicate channel is removed and replaced according to the new priority. If the client's configuration priorities are going to be reordered by an action, the {webui} requires you to confirm the change before proceeding.

Compare Configuration Files

You can also use the system set manager (SSM) to compare the configuration file deployed on clients with the configuration file stored on the Uyuni Server.

Procedure: Comparing Configuration Files

1. In the Uyuni {webui}, navigate to menu:Systems[Systems List] and select the clients subscribed to the configuration files you want to compare.

2. Navigate to menu:Systems[System Set Manager], and go to the menu:Configuration[Compare Files] subtab to the list of available configuration files.

3. OPTIONAL: Click the number in the `Systems` column to see which clients are currently subscribed to the configuration file.

4. Check the configuration files to compare, and click btn:[Schedule File Comparison].

Configuration File Macros on Traditional Clients

Being able to store one file and share identical configurations is useful, but in some cases you might need many variations of the same configuration file, or configuration files that differ only in system-specific details, such as host name and MAC address. In this case, you can use macros or variables within the configuration files. This allows you to upload and distribute a single file, with hundreds or even thousands of variations. In addition to variables for custom system information, the following standard macros are supported:

```
rhn.system.sid
rhn.system.profile_name
rhn.system.description
rhn.system.hostname
rhn.system.ip_address
rhn.system.custom_info(key_name)
rhn.system.net_interface.ip_address(eth_device)
rhn.system.net_interface.netmask(eth_device)
rhn.system.net_interface.broadcast(eth_device)
rhn.system.net_interface.hardware_address(eth_device)
rhn.system.net_interface.driver_module(eth_device)
```

To use this feature, either upload or create a configuration file via the `Configuration Channel Details` page. Then open its `Configuration File Details` page and include the supported macros of your choice. Ensure that the delimiters used to offset your variables match those set in the `Macro Start Delimiter` and `Macro End Delimiter` fields and do not conflict with other characters in the file. We recommend that the delimiters be two characters in length and do not contain the percent (`%`) symbol.

For example, you may have a file applicable to all of your servers that differs only in IP address and host name. Rather than manage a separate configuration file for each server, you can create a single file, such as `server.conf`, with the IP address and host name macros included.

```
hostname={| rhn.system.hostname |}
ip_address={| rhn.system.net_interface.ip_address(eth0) |}
```

When the file is delivered to individual systems, whether through a scheduled action in the Uyuni {webui} or at the command line with the Uyuni Configuration Client (`mgrcfg-client`), the variables will be replaced with the host name and IP address of the system as recorded in Uyuni's system profile. In this example, the deployed version will look similar to this:

```
hostname=test.example.domain.com
ip_address=177.18.54.7
```

To capture custom system information, insert the key label into the custom information macro (`rhn.system.custom_info`). For example, if you developed a key labeled "asset" you can add it to the custom information macro in a configuration file to have the value substituted on any system containing it. The macro would look like this:

```
asset={@ rhn.system.custom_info(asset) @}
```

When the file is deployed to a system containing a value for that key, the macro gets translated, resulting in a string similar to this:

```
asset=Example#456
```

To include a default value (for example, if one is required to prevent errors), you can append it to the custom information macro, like this:

```
asset={@ rhn.system.custom_info(asset) = 'Asset #' @}
```

This default is overridden by the value on any system containing it.

The Uyuni Configuration Manager (`mgrcfg-manager`) is available on Uyuni client machines to assist with system management. It will not translate or alter files, as the tool is system agnostic. The `mgrcfg-manager` command does not depend on system settings. Binary files cannot be interpolated.

Power Management

You can power on, power off, and reboot clients using the Uyuni {webui}.

This feature uses either the IPMI or Redfish protocol and is managed using a Cobbler profile. The selected client must have a power management controller supporting one of these protocols.

For Redfish, ensure you can establish a valid SSL connection between the client and the Uyuni Server. You must have trusted the certificate authority used to sign the SSL Server Certificate of the Redfish management controller. The CA certificate must be in `.pem` format, and stored on the Uyuni Server at `/etc/pki/trust/anchors/`. When you have saved the certificate, run `update-ca-certificate`.

Procedure: Enabling Power Management

1. In the Uyuni {webui}, navigate to menu:Systems[Systems List], select the client you want to manage, and navigate to the menu:Provisioning[Power Management] tab.

2. In the `Type` field, select the power management protocol to use.

3. Complete the details for the power management server, and click the appropriate button for the action to take, or click btn:[Save only] to save the details without taking any action.

You can apply power management actions to multiple clients at the same time by adding them to the system set manager. For more information about using the system set manager, see **[Client-configuration › System-set-manager ›]**.

Power Management and Cobbler

The first time you use a power management feature, a Cobbler system record is automatically created, if one does not yet exist for the client. These automatically created system records are not bootable from the network, and include a reference to a dummy system image. This is needed because Cobbler does not currently support system records without profiles or images.

Cobbler power management uses fence-agent tools to support protocols other than IPMI. Only IPMI and Redfish protocols are supported by Uyuni. You can configure your client to use other protocols by adding the fence-agent names as a comma-separated list to the `java.power_management.types` configuration parameter in the `rhn.conf` configuration files.

Configuration Snapshots

Snapshots record the package profile, configuration files, and Uyuni settings for a client at a set point in time. You can roll back to older snapshots to restore previous configuration settings.

NOTE Snapshots are supported on traditional clients only. Salt clients do not support this feature.

Snapshots are captured automatically after some actions occur. You can also manually take a snapshot at any time. We recommend that you ensure you have a current snapshot before performing any potentially destructive action on your clients.

Snapshots are enabled by default. You can disable automatic snapshots by setting `enable_snapshots=0` in the `rhn.conf` configuration file.

Manage your snapshots by navigating to menu:Systems[Systems List] and selecting the client you want to manage. Navigate to the menu:Provisioning[Snapshots] tab to see a list of all current snapshots for the selected client. Click the name of a snapshot to see more information about the changes recorded in the snapshot. You can use the subtabs in the menu:Provisioning[Snapshots] tab to see the changes that rolling back to the selected snapshot makes to:

- Group memberships
- Channel subscriptions
- Installed packages
- Configuration channel subscriptions
- Configuration files
- Snapshot tags

IMPORTANT You can use a snapshot to roll back most changes to a client, but not all of them. For example, you cannot roll back multiple updates, and you cannot roll back a product migration. Always ensure you have taken a backup before performing upgrades on your clients.

Snapshot Tags

Snapshot tags allow you to add meaningful descriptions to your snapshots. You can use tags to record extra information about snapshots, such as a last known working configuration, or a successful upgrade.

Manage your snapshot tags by navigating to menu:Systems[Systems List] and selecting the client you want to manage. Navigate to the menu:Provisioning[Snapshot Tags] tab to see a list of all current snapshot tags for the selected client. Click `Create System Tag`, enter a description, and click the btn:[Tag Current Snapshot] button.

Snapshots on Large Installations

There is no maximum number of snapshots that Uyuni keeps. This means that the database that stores the snapshots grows as you add more clients, packages, channels, and configuration changes.

If you have a large installation, with thousands of clients, you can use the Uyuni API to create a recurring cleanup script on a recurring schedule to ensure that old snapshots are deleted regularly. Alternatively, you can disable the feature by setting `enable_snapshots=0` in the `rhn.conf` configuration file.

Custom System Information

You can include customized system information about your clients. System information is defined as key:value pairs, which can be assigned to clients. For example, you can define a key:value pair for a particular processor, then assign that key to all clients that have that processor installed. Custom system information is categorized, and can be searched using the Uyuni {webui}.

Before you begin, you need to create a key that allows you to store custom information.

Procedure: Creating a Custom System Information Key

1. In the Uyuni {webui}, navigate to menu:Systems[Custom System Info], and click btn:[Create Key].

2. In the `Key Label` field, add a name for your key. Do not use spaces. For example, `intel-x86_64-quadcore`.

3. In the `Description` field, provide any additional information required.

4. Repeat for each key you require.

For traditional clients, this information is stored in the Uyuni database. For Salt clients, this information is stored in a Salt pillar. You can retrieve this information from a Salt client with a command such as:

```
salt $minionid pillar.get custom_info:key1
```

This command will result in an output such as:

```
$minionid: val1
```

When you have created some custom system information keys, you can apply them to clients.

Procedure: Applying Custom Information Keys to Clients

1. In the Uyuni {webui}, navigate to `Systems`, click the client to apply custom information to, and navigate to the menu:Details[Custom Info] tab.

2. Click btn:[Create Value].

3. Locate the value you want to apply, and click the key label.

4. In the `Value` field, provide any additional information.

5. Click btn:[Update Key] to apply the custom information to the client.

For more information about configuration management, see **[Client-configuration › Configuration-management ›]**.

Client Upgrades

Clients use the versioning system of their underlying operating system, and require regular upgrades.

For SCC registered clients using {suse} operating systems, you can perform upgrades within the Uyuni {webui}. For supported {sle} 15 upgrade paths, see https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-upgrade-paths.html

To upgrade clients running SLE 12 to SLE 15, the upgrade is automated, but you need to do some preparation steps before you begin. For more information, see **[Client-configuration › Client-upgrades-major ›]**.

You can also automate client upgrades using the content lifecycle manager. For more information, see **[Client-configuration › Client-upgrades-lifecycle ›]**.

For more information about service pack upgrades (product migration), see **[Client-configuration › Client-upgrades-product-migration ›]**.

For more information about upgrading unregistered openSUSE Leap clients, see **[Client-configuration › Client-upgrades-uyuni ›]**.

Client - Major Version Upgrade

Your clients must have the latest available service pack (SP) for the installed operating system, with all the latest updates applied. Before you start, ensure that the system is up to date and all updates have been installed successfully.

The upgrade is controlled by {yast} and {ay}, it does not use Zypper.

Prepare to Migrate

Before you can migrate your client from SLE 12 to SLE 15 , you need to:

1. Prepare installation media

2. Create an auto-installable distribution

3. Create an activation key

4. Upload an {ay} profile

Procedure: Preparing Installation Media

1. On the Uyuni Server, create a local directory for the SLE 15 SP2 installation media:

   ```
   mkdir -p /srv/images/sle15sp2
   ```

2. Download an ISO image with the installation sources, and mount the ISO image on your server:

```
mount -o loop DVD1.iso /mnt/
```

3. Copy everything from the mounted ISO to your local file system:

```
cp -r /mnt/* /srv/images/sle15sp2
```

4. When the copy is complete, unmount the ISO image:

```
umount /mnt
```

NOTE This image is the unified installer and can be used for multiple autoinstallable distributions.

Procedure: Creating an Autoinstallable Distribution

1. In the Uyuni {webui}, navigate to menu:Systems[Autoinstallation > Distributions] and click btn:[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. For example, `sles15sp2-x86_64`.

   ◦ In the `Tree Path` field, type an absolute path to the installation source. For example, `/srv/images/sle15sp2`.

   ◦ In the `Base Channel` field, select `SLE-Product-SLES15-SP2-Pool for x86_64`.

   ◦ In the `Installer Generation` field, select `SUSE Linux Enterprise 15`.

   ◦ 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 btn:[Create Autoinstallable Distribution] to save.

To switch from the old SLE 12 base channel to the new SLE 15 channel, you need an activation key.

Procedure: Creating an Activation Key

1. In the Uyuni Server {webui}, navigate to menu:Systems[Activation Keys] and click `Create Key`.

2. Enter a description for your key.

3. Enter a key or leave it blank to generate an automatic key.

4. OPTIONAL: If you want to limit the usage, enter your value in the `Usage` text field.

5. Select the `SLE-Product-SLES15-SP2-Pool for x86_64` base channel.

6. OPTIONAL: Select any `Add-On System Types`. For more information, see https://documentation.suse.com/sles/15-SP3/html/SLES-all/article-modules.html.

7. Click btn:[Create Activation Key].

8. Click the `Child Channels` tab and select the required channels.

9. Click btn:[Update Key].

Create 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. For example scripts that you can use as a starting point, see https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST. For valid AutoYaST upgrade settings, see https://doc.opensuse.org/projects/autoyast/#CreateProfile-upgrade.

Procedure: Creating an Autoinstallation Profile

1. In the Uyuni {webui}, navigate to menu:Systems[Autoinstallation > Profiles] and upload your autoinstallation profile script. For example scripts that you can use as a starting point, see https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST.

2. In the `Kernel Options` field, type `autoupgrade=1`. Optionally, you can also include the `Y2DEBUG=1` option. The debug setting is not required but can help with investigating any future problems you might encounter.

> **IMPORTANT** Clients running in Azure cloud must add `textmode=1 console=ttyS0` to `Kernel Options`.

3. Paste the autoinstallation profile or use the file upload field.

4. Click btn:[Create] to save.

5. When the uploaded profile requires variables to be set, navigate to menu:Systems[Autoinstallation > Profiles], select the profile to edit, and navigate to the `Variables` tab. Specify the required variables, using this format:

```
<key>=<value>
```

Migration

Before you begin, check that all the channels referenced in the autoinstallation profile are available and fully synchronized.

You can monitor the mirroring progress in `/var/log/rhn/reposync/<channel-label>.log`.

Procedure: Migrating

1. In the Uyuni Server {webui}, navigate to `Systems` and select the client to be upgraded.

2. Navigate to the `Provisioning` tab, and select the autoinstallation profile you uploaded.

3. Click btn:[Schedule Autoinstallation and Finish]. The system downloads the required files, change the bootloader entries, reboot, and start the upgrade.

Next time the client synchronizes with the Uyuni Server, it receives a re-installation job. The re-installation job fetches the new kernel and initrd packages. It also writes a new `/boot/grub/menu.lst`, containing pointers to the new kernel and initrd packages.

When the client next boots, it uses grub to boot the new kernel with its initrd. PXE booting is not used during this process.

Approximately three minutes after the job was fetched, the client goes down for reboot.

> **NOTE**　For Salt clients, use the `spacewalk/minion_script` snippet to register the client again after migration has completed.

Upgrade Using the Content Lifecycle Manager

When you have many {sles} clients to manage, you can automate in-place upgrades using the content lifecycle manager.

Prepare to Upgrade

Before you can upgrade your clients, you need to make these preparations:

- Create a content lifecycle project

- Create an activation key

- Create an autoinstallable distribution

- Create an autoinstallation profile

Procedure: Creating a Content Lifecycle Project

1. Create a content lifecycle project for your distribution. For more information, see **[Administration › Content-lifecycle ›]**.

2. Ensure you a choose a short but descriptive name for your project.

3. Include all source channel modules that you require for your distribution.

4. Add filters as required, and set up at least one environment.

Procedure: Creating an Activation Key

1. Create an activation key for your distribution. For more information, see **[Client-configuration › Activation-keys ›]**.

2. Ensure your activation key includes all filtered project channels.

Procedure: Creating an Autoinstallable Distribution

1. Create an autoinstallable distribution for every base channel you want to migrate. For more information, see **[Client-configuration › Autoinst-distributions ›]**.

2. Give your distribution a label that references the name of the content lifecycle project.

3. In the `Installer Generation` field, select the SLES version you are using.

Procedure: Creating an Autoinstallation Profile

1. Create an autoinstallation profile for every target distribution and service pack you want to upgrade to. For more information, see **[Client-configuration › Autoinst-profiles ›]**.

2. You must use a different profile for Salt and traditional clients.

3. You can use variables in the profile to distinguish between the different lifecycle environments.

For example autoinstallation profiles, see https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST.

Use these variables in your autointallation profiles for automating in-place upgrades:

Example: Variables for Use in Autoinstallation Profiles

```
registration_key=1-15sp1-demo-test
org=1
channel_prefix=15sp1-demo-test
distro_label=15sp1-demo-test
```

Example: Entry for Use in Autoinstallation Profiles

```
<listentry>
    <ask_on_error config:type="boolean">true</ask_on_error>
    <media_url>https://$redhat_management_server/ks/dist/child/$channel_prefix-sle-module-
web-scripting15-sp1-pool-x86_64/$distro_label</media_url>
    <name>$channel_prefix SLE-Module-Web-Scripting15-SP1 Pool for x86_64 </name>
    <product>Web Scripting Module 15 SP1 x86_64 Pool</product>
</listentry>
```

Upgrade

When you have prepared the server for the upgrade, you can provision the clients.

Procedure: Provisioning the Clients

1. In the Uyuni {webui}, navigate to menu:Systems[System List], and select the clients you want to provision to add them to the system set manager.

2. Navigate to menu:Systems[System Set Manager > Overview] and click the `Provisioning` tab.

3. Select the autoinstallation profile to use.

For clients that are able to use PXE, the migration is automated as soon as you have provisioned them.

For all other clients, you can use Cobbler to perform the upgrade.

Procedure: Using Cobbler to Upgrade Clients

1. At the command prompt, as root, check the available Cobbler profiles:

```
cobbler profile list
```

2. Build the ISO file with your chosen profile:

```
cobbler buildiso --iso=/tmp/SLE_15-sp1.iso --profiles=SLE_15-sp1:1:Example
```

For more information about using CD-ROMs to provision clients, see **[Client-configuration › Autoinst-cdrom ›]**.

Product Migration

Product migration allows you to upgrade SLE-based client systems from an Service Pack (SP) level to a later one. For example, you can migrate {sle} 15 SP1 to {sle} 15 SP2.

You can also migrate openSUSE Leap to a later minor version or to the corresponding SLE SP level, for example:

- openSUSE Leap 15.1 to 15.2, or

- openSUSE Leap 15.1 to {sle} 15 SP1, or

- openSUSE Leap 15.2 to {sle} 15 SP2

WARNING
During migration, Uyuni automatically accepts any required licenses (EULAs) before installation.

In {sle} 12 and later, {suse} supports service pack skipping if {scc} provides it. For example, you can upgrade from {sle} 15 to SP2, without installing SP1. For supported {sle} upgrade paths, see https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-upgrade-paths.html#sec-upgrade-paths-supported.

IMPORTANT
Product migration is for upgrading within the same major version. You cannot use product migration to migrate from {sle} 12 to {sle} 15. For more information about major upgrades, see **[Client-configuration › Client-upgrades-major ›]**.

WARNING
Product migration does not have a rollback feature. When the migration procedure has begun, rolling back is not possible. Ensure you have a working system backup available, in case of an emergency.

Before starting the product migration, ensure there are no pending updates or patches. Check the `System Status` on the client system's menu:Details[Overview] page, and install all offered updates or patches. If your client system is not uptodate, product migration may fail.

NOTE	Before starting a migration, make sure all the channels of the target product are fully synchronized. To check the synchronization status in the {webui}, navigate to the menu:Admin[Setup Wizard > Products] page.

Procedure: Performing a Migration

1. From the menu:Systems[Overview] page, select a client.

2. From the system details page of the client, navigate to the menu:Software[Product Migration] tab.

3. Select the target migration path and click btn:[Select Channels].

4. From the `Product Migration - Channels` page select the correct base channel, including `Mandatory Child Channels` and any additional `Optional Child Channels`.

5. OPTIONAL: Check `Allow Vendor Change` to allow packages that have changed vendors to be installed. If this occurs, a notification is shown with details before the migration is started.

IMPORTANT	To migrate openSUSE Leap to {sle}, you must check the `Allow Vendor Change` option.

6. Click btn:[Schedule Migration] when your channels have been configured properly.

Product Mass Migration

If you want to migrate a large number of clients to the next SP version, you can use Uyuni API calls.

Procedure: Performing a Product Mass Migration

1. List available migration targets, and take note of the system IDs you want to migrate:

```
spacecmd api -- system.listMigrationTargets -A 1000010001
```

2. For each system ID, call `listMigrationTarget` and check that the desired target product is available.

   ◦ If the system ID has an available target, call `system.scheduleProductMigration`.

   ◦ If the desired target is not available, skip the system.

Adapt this template for your environment:

```
target = '[....]'
basechannel = 'channel-label'
system_ids = [1, 2, 3]

session = auth.login(user, pass)
for system in system_ids
  if system.listMigrationTargets(session, system).ident == target
    system.scheduleProductMigration(session, system, target, basechannel, [], False, <now>)
  else
    print "Cannot migrate to requested target -- skipping system"
  endif
endfor
```

Upgrade Uyuni Clients

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 \
   opensuse_leap15_2 \
   opensuse_leap15_2-non-oss \
   opensuse_leap15_2-non-oss-updates \
   opensuse_leap15_2-updates \
   opensuse_leap15_2-uyuni-client
   ```

2. Fully synchronize all channels with `spacewalk-repo-sync`. In case of already defined repository URLs, continue with upgrade:proxy-uyuni.pdf.

3. In the Uyuni Server {webui}, navigate to menu:Software[Manage > Channels] and click the `Uyuni Client Tools for openSUSE Leap 15.2 (x86_64)` channel name.

4. In the upper right corner, click btn:[Manage Channel].

5. Click the `Repositories` tab, and select `External - Uyuni Client Tools for openSUSE Leap 15.1 (x86_64)`.

6. Click btn:[Update Repositories].

7. Navigate to menu:Repositories[Sync] subtab, and click btn:[Sync Now].

8. Do the same with `openSUSE Leap 15.2 (x86_64)` and `External - openSUSE Leap 15.1 (x86_64)`.

Unfold `openSUSE Leap 15.2 (x86_64)` to see all child channels populated with packages.

Upgrade the Client

To upgrade a client you replace the software repositories and update the software, and finally reboot the client.

Procedure: Updating the Client

1. In the Uyuni Server {webui}, navigate to menu:Systems[] and click the name of the client.

2. Click menu:Software[Software Channels], and as the base channel select the openSUSE Leap {opensuse-version} channel that is listed in the `Customs Channels` list.

3. In the `Child Channels` pane, select the {opensuse-version} child channels.

4. Click btn:[Next], and `Confirm Software Channel Change` with btn:[Confirm].

5. Click menu:Software[Packages > Upgrade], and select all the packages to be updated on the client, and then apply the selection. Click btn:[Upgrade Packages], check the details, and click btn:[Confirm] to complete the update.

6. Reboot the client.

If you need to update many clients, you can create an action chain of this command sequence on the Uyuni Server. You can use the action chain to perform updates on multiple clients at the same time.

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 {ay} or {kickstart}, and manage guests in the {webui}.

Additionally, {caasp} clusters can be managed in the Uyuni {webui}, by navigating to menu:Clusters[Overview].

For VMWare, including VMWare vSphere, and Nutanix AHV, 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.

For more information about using VHMs to manage virtualization, see **[Client-configuration › Vhm ›]**.

Manage Virtualized Hosts

Before you begin, ensure that the client you want to use as a virtualization host has the `Virtualization Host` system type assigned to it. Both traditional and Salt clients can be used as virtual hosts. Navigate to menu:Systems[Systems List] and click the name of the client to use as a virtualization host. The system types are listed in the `System Properties` section. If the `Virtualization Host` system type is not listed, click btn:[Edit These Properties] to assign it.

When a client has the `Virtualization Host` system type, the `Virtualization` tab is available in the System Details page for the client. The `Virtualization` tab allows you to create and manage virtual guests, and manage storage pools and virtual networks.

Create Virtual Guests

You can add virtual guests to your virtualization hosts within the Uyuni {webui}.

Procedure: Creating a Virtual Guest

1. In the Uyuni {webui}, navigate to menu:Systems[Systems List], click the name of the virtualization host, and navigate to the `Virtualization` tab.

2. In the `General` section, complete these details:

- ◦ In the `Guests` subtab, click btn:[Create Guest].

- ◦ In the `Name` field, type the name of the guest.

- ◦ In the `Hypervisor` field, select the hypervisor to use.

- ◦ In the `Virtual Machine Type` field, select either fully virtualized or para-virtualized.

- ◦ In the `Maximum Memory` field, type the upper size limit for the guest disk, in MiB.

- ◦ In the `Virtual CPU count`, type the number of vCPUs for the guest.

- ◦ In the `Architecture` field, select the emulated CPU architecture to use on the guest. By default, the architecture selected matches the virtual host.

- ◦ In the `Auto-installation Profile` field, select the auto-installation tool to use to install the guest. Leave this field blank if you do not want to use auto-installation.

3. In the `Disks` section, complete the details of the virtual disk to use with the client. In the `Source template image URL` field, ensure you type the path to an operating system image. If you do not do this, your guest is created with an empty disk.

4. In the `Networks` section, complete the details of the virtual network interface to use with the client. Leave the `MAC address` field blank to generate a MAC address.

5. In the `Graphics` section, complete the details of the graphics driver to use with the client.

6. Schedule a time for the guest to be created, and click btn:[Create] to create the guest.

7. The new virtual guest starts as soon as it has successfully been created.

You can add virtual guests on a pacemaker cluster within the Uyuni {webui}, too.

Procedure: Creating a cluster managed Virtual Guest

1. Follow the `Creating a Virtual Guest` procedure on one of the nodes of the cluster with the following additions:

   - ◦ Ensure the `Define as a cluster resource` field is checked.

   - ◦ In the `Path to the cluster shared folder for VM definitions` field, type the path to a folder shared by all cluster nodes where the guest configuration will be stored.

   - ◦ Make sure every disk is located on a storage pool shared by all the cluster nodes.

Virtual Guests managed by a cluster can be live migrated.

Clusters

You can use Uyuni to directly manage {caasp} clusters. Manage clusters that you have registered with Uyuni by navigating to menu:Clusters[Overview] in the {webui}.

Autoinstallation Profile

{caasp} 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 {caasp} clusters.

> **NOTE** Only {caasp} 4 is currently supported.

Before you begin, ensure you have installed your {caasp} cluster.

Elect a Management Node

To manage a {caasp} 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 {caasp} 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 {webui}, navigate to menu:Systems[System List] and click the name of the client to elect as the management node.

2. Navigate to the menu:Formulas[Configuration] tab, and check the `CaaSP Management Node` formula.

3. Click btn:[Save] and apply the highstate.

> **NOTE** Do not use the management node until the highstate has been completed.

List all known clusters by navigating to menu: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 {kube} 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 {webui}.

2. Provide a way to authenticate.

There are two ways you can authenticate a {caasp} 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 {caasp}. 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.

> **NOTE**
>
> 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 {webui}. 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 {webui}.

Procedure: Adding an Existing Cluster

1. In the Uyuni {webui}, navigate to menu:Clusters[Overview] and click btn:[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 btn:[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 menu:Clusters[Overview], unchecking the cluster to be deleted, and clicking btn:[Delete Cluster].

> **IMPORTANT**   Deleting a cluster removes the cluster from Uyuni, it does not delete the cluster nodes. Workloads running on the cluster 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 {caasp} 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 {caasp} 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 {webui}, navigate to menu:Clusters[Overview] and click btn:[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 {caasp} 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 btn:[Save] to schedule an action to add the nodes to the {caasp} cluster.

Procedure: Removing Nodes from a Cluster

1. In the Uyuni {webui}, navigate to menu:Clusters[Overview], uncheck the nodes to remove, and click btn:[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 btn:[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 {webui}, navigate to menu:Clusters[Overview], and click the cluster to upgrade.

2. OPTIONAL: The are no {caasp} 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 btn:[Save] to schedule an action to upgrade the cluster.

> **NOTE** Uyuni only interacts 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.

Virtualization with Xen and KVM

Xen and KVM virtualized clients can be managed directly in Uyuni.

To begin, you need to set up a virtual host on your Uyuni Server. You can then set up autoinstallation using {ay} 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-SP3/html/SLES-all/book-virtualization.html.

For {rhel} operating systems, refer to the Red Hat documentation for your version.

Uyuni uses `libvirt` to install and manage guests. You must have the `libvirtd` 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 need to perform some configuration changes. For more information about how to set this up, consult the relevant

documentation for your operating system.

You need 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 need to enable the `Virtualization Host` entitlement. This allows you to see VM changes instantly. To do this, in the Uyuni {webui}, 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 btn:[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, 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 executes 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 {ay} or {kickstart} to automatically install and register Xen and KVM guests.

You need 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 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 {ay} or {kickstart}.
IMPORTANT	

Create an Autoinstallable Distribution

You 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 differs depending on whether you are using a SLES or {rhel} operating system on your guests. The packages for a {rhel} 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 41. Paths for autoinstallable distributions

Operating System Type	Kernel Location	initrd Location
{rhel}	`images/pxeboot/vmlinuz`	`images/pxeboot/initrd.img`
SLES	`boot/<arch>/loader/initrd`	`boot/<arch>/loader/linux`

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 {webui}, navigate to menu:Systems[Autoinstallation > Distributions] and click btn:[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 btn:[Create Autoinstallable Distribution] to save.

When you have created an autoinstallable distribution, you can edit it by navigating to menu: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 {webui}, by navigating to menu:Systems[Autoinstallation > Profiles], clicking btn:[Create New Kickstart File], and following the prompts.

You can also create {ay} or {kickstart} autoinstallation profiles by hand. {suse} provides templates of {ay} installation files that you can use as a starting point for your own custom files. You will find them at https://github.com/SUSE/manager-build-profiles.

If you are using {ay} to install SLES, you also need to include this snippet:

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

- For more on {ay}, see client-configuration:autoinst-profiles.pdf.

- For more on {kickstart}, see client-configuration:autoinst-profiles.pdf, or refer to the Red Hat documentation for your installation.

Procedure: Uploading an Autoinstallation Profile

1. In the Uyuni {webui}, navigate to menu:Systems[Autoinstallation > Profiles] and click btn:[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 btn:[Choose File], and use the system dialog to select the file to upload. If the file is successfully uploaded, the filename is shown in the `File to Upload` field.

- The contents of the uploaded file is shown in the `File Contents` field. If you need to make edits, you can do so directly.

3. Click btn:[Create] to save your changes and store the profile.

When you have created an autoinstallation profile, you can edit it by navigating to menu:Systems[Autoinstallation > Profiles] and selecting the profile you want to edit. Make the desired changes and save your settings by clicking btn:[Create].

IMPORTANT	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 {ay} 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 {ay} 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 {webui}, navigate to menu:Systems[Autoinstallation > Profiles], and select the {ay} 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 menu:Update[] to save your changes.

	If your {ay} profile already contains a `<scripts>` section, do not add a
IMPORTANT	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
IMPORTANT	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 is canceled.

1. In the Uyuni {webui}, navigate to menu: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 btn:[Advanced Options].

6. Click btn:[Schedule Autoinstallation and Finish] to complete.

Manage VM Guests

You can use the Uyuni {webui} to manage your VM Guests, including actions like shutting down, restarting, and adjusting CPU and memory allocations.

To do this, you need your Xen or KVM VM host registered to the Uyuni Server, and have the `libvirtd` service running on the host. For traditional clients, you also need the `mgr-cfg-actions` package installed on your Uyuni Server.

In the Uyuni {webui}, navigate to menu: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 {webui}, see **[Reference › Systems ›]**.

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{ {webui}. 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 {kube} and {caasp}.

- For more information on using a VHM with Amazon Web Services, see **[Client-configuration › Vhm-aws ›]**.

- For more information on using a VHM with Microsoft Azure, see **[Client-configuration › Vhm-azure ›]**.

- For more information on using a VHM with {caasp}, see **[Client-configuration › Vhm-caasp ›]**.

- For more information on using a VHM with Google Compute Engine, see **[Client-configuration › Vhm-gce ›]**.

- For more information on using a VHM with {kube}, see **[Client-configuration › Vhm-kubernetes ›]**.

- For more information on using a VHM with Nutanix, see **[Client-configuration › Vhm-nutanix ›]**.

- For more information on using a VHM with VMWare vSphere, see **[Client-configuration › Vhm-vmware ›]**.

- For more information on using a VHM with other hosts, see **[Client-configuration › Vhm-file ›]**.

VHM and Amazon Web Services

You can use a virtual host manager (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 {webui}, navigate to menu:Systems[Virtual Host Managers].

2. Click btn:[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:virtualization.pdf.

4. Click btn:[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 btn:[Refresh Data] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to menu:Systems[System List > Virtual Systems].

Instances running on the Amazon public cloud 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 virtual host manager (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 {webui}, navigate to menu:Systems[Virtual Host Managers].

2. Click btn:[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 btn:[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 btn:[Refresh Data] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to menu: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 report this UUID to the Uyuni Server:

```
13f56399-bd52-4150-9748-7190aae1ff21
```

VHM and {caasp}

You can use a virtual host manager (VHM) to manage {caasp} clusters. The VHM allows Uyuni to obtain and report information about your clusters. For more information on VHMs, see **[Client-configuration › Vhm ›]**.

| NOTE | You can also manage {caasp} clusters directly with Uyuni, without using a VHM. For more information, see **[Client-configuration › Virt-clusters ›]**. |

Onboarding CaaSP Nodes

You can register each {caasp} 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 {caasp} 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 {caasp} nodes to Uyuni, the registered system automatically applies the system lock Salt formula to prevent unintended actions on the client. When a system is locked, the {webui} shows a warning and you can schedule actions using the {webui} or the API, but the action fails. 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 {kube} are managed using the `skuba-update` tool. For more information, see https://documentation.suse.com/suse-caasp/4/html/caasp-admin.

WARNING

When using Salt or Uyuni (either via UI or API) on any {caasp} 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 a product 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 {caasp} cluster unusable. Uyuni does not stop you from issuing these operations if the system is not locked.

Autoinstallation Profile of a {caasp} 4 Node

{caasp} 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 based on the {caasp} 4 template, customized to make use of Uyuni, see https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST/CaaSP-autoinstall.

Manage a {caasp} Cluster With Uyuni

You can use Uyuni to manage one or more existing {caasp} clusters.

NOTE   Only {caasp} 4 is currently supported.

Before you begin, ensure you have installed your {caasp} cluster.

Elect a Management Node

To manage a {caasp} 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 {caasp} 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 {webui}, navigate to menu:Systems[System List] and click the name of the client to elect as the management node.

2. Navigate to the menu:Formulas[Configuration] tab, and check the `CaaSP Management Node` formula.

3. Click btn:[Save] and apply the highstate.

> **NOTE**   You cannot use the management node until the highstate has been completed.

List all known clusters by navigating to menu: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 {kube} 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 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 {webui}.

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 {caasp}:

      ▪ 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.

> **NOTE**
>
> 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 {webui}. 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 {webui}.

Procedure: Adding an Existing Cluster

1. In the Uyuni {webui}, navigate to menu:Clusters[Overview] and click btn:[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 btn:[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 menu:Clusters[Overview], unchecking the cluster to be deleted, and clicking btn:[Delete Cluster].

> **IMPORTANT**
>
> Deleting a cluster removes the cluster from Uyuni, it does not delete the cluster nodes. Workloads running on the cluster 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 {caasp} channel

assigned.

1. In the Uyuni {webui}, navigate to menu:Clusters[Overview] and click btn:[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 {caasp} 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 btn:[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 {webui}, navigate to menu:Clusters[Overview], uncheck the nodes to remove, and click btn:[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 btn:[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 {webui}, navigate to menu:Clusters[Overview], and click the cluster to upgrade.

2. OPTIONAL: The are no {caasp} 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 btn:[Save] to schedule an action to upgrade the cluster.

> **NOTE** Uyuni only interacts 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.

VHM and Google Compute Engine

You can use a virtual host manager (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 needs to have port 443 open, 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 {webui}, navigate to menu:Systems[Virtual Host Managers].

2. Click btn:[Create] and select `Google Compute Engine` from the drop-down menu.

3. In the `Add a Google Conpute 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 btn:[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 btn:[Refresh Data] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to menu: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 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 report this UUID to Uyuni Server:

```
152986662232938449
```

VHM and Kubernetes

You can use a virtual host manager (VHM) to manage {kube} 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 {kube}, 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 also require:

- At least one {kube} or {caasp} cluster available on your network.

- The `virtual-host-gatherer-Kubernetes` package installed on the Uyuni Server.

- {kube} version 1.5.0 or higher, or {caasp}.

- Docker version 1.12 or higher on the container build host.

Create a {kube} VHM

{kube} clusters are registered with Uyuni as a VHM.

You will need a `kubeconfig` file to register and authorize your {kube} cluster. You can get a `kubeconfig` file using the {kube} command line tool `kubectl`.

If you are using {caasp}, you can download the file from the Velum interface.

Procedure: Creating a {kube} VHM

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

2. Click btn:[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 {kube} cluster.

4. In the `context` field, select the appropriate context for the cluster. This is specified in the `kubeconfig` file.

5. Click btn:[Create].

Procedure: Viewing the Nodes in a Cluster

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

2. Select the {kube} cluster.

3. Refresh the node data by clicking btn:[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`.

> **NOTE** Node data is not refreshed during registration. You need to manually refresh the data to see it.

Retrieve Image Runtime Data

You can view runtime data about {kube} images in the Uyuni {webui}, by navigating to

menu: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 {webui}, navigate to menu:Images[Stores].

2. Click btn:[Create] to create an image store.

3. Navigate to menu:Images[Profiles].

4. Click btn:[Create] to create an image profile. You will need to use a dockerfile that is suitable to deploy to {kube}.

5. Navigate to menu:Images[Build] to build an image with the new profile.

6. Deploy the image into one of the registered {kube} clusters. You can do this with the **kubectl** tool.

The updated data should now be available in the image list at menu:Images[Image List].

Procedure: Importing a Previously Deployed Image

1. In the Uyuni {webui}, navigate to menu:Images[Image Stores].

2. Add the registry that owns the image you want to import, if it is not already there.

3. Navigate to menu:Images[Image List] and click btn:[Import].

4. Complete the fields, select the image store you created, and click btn:[Import].

The imported image should now be available in the image list at menu:Images[Image List].

Procedure: Rebuilding a Previously Deployed Image

1. In the Uyuni {webui}, navigate to menu:Images[Image List], locate the row that contains the image you want to rebuild, and click btn:[Details].

2. Navigate to the `Build Status` section, and click btn:[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 menu:Images[Image List]. This shows that the instances are running a previous build of the image.

> **NOTE**
> 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 {webui}, navigate to menu:Images[Image List], locate the row that contains the running instance, and click btn:[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 {kube} 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

> **IMPORTANT**
> 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 reponse, the entire cluster is ignored by Uyuni.

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

Virtualization with Nutanix

You can use Nutanix AHV virtual machines with Uyuni by setting up a virtual host manager (VHM). To begin, you need to set up a VHM on your Uyuni Server, and inventory the available VM hosts.

VHM Setup

The Virtual Host Manager (VHM) runs on the Uyuni Server.

Ensure you have installed the `virtual-host-gatherer-Nutanix` package on the Uyuni Server.

To run a VHM, your Uyuni Server must have port 9440 open to access the Nutanix Prism Element API.

Procedure: Creating a Nutanix VHM

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

2. Click btn:[Create] and select `Nutanix AHV`.

3. In the `Add a Nutanix AHV 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 Prism Element API port to use (for example, `9440`).

   - 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 btn:[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 btn:[Refresh Data] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to menu:Systems[System List > Virtual Systems].

NOTE
Connecting to the Nutanix Prism API server from a browser using HTTPS can sometimes log an `invalid certificate` error. If this occurs, refreshing the data from the virtual host manager fails. A valid SSL certificate (not self-signed) is required on your Nutanix API server. If you're using a custom CA authority for your Nutanix SSL certificate, copy the custom CA certificate to `/etc/pki/trust/anchors` on the Uyuni Server. 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 runs data collection automatically. If you want to manually perform data collection, navigate to menu:Systems[Virtual Host Managers], select the appropriate VHM, and click btn:[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/gatherer.log`.

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 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 needs to have port 443 open, 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 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 {webui}, navigate to menu:Systems[Virtual Host Managers].

2. Click btn:[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 btn:[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 btn:[Refresh Data] to inventory the new VHM.

To see which objects and resources have been inventoried, navigate to menu:Systems[System List > Virtual Systems].

NOTE	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 fails. 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 runs data collection automatically. If you want to manually perform data collection, navigate to menu:Systems[Virtual Host Managers], select the appropriate VHM, and click btn:[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/gatherer.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.

1. Download the CA Certificate from your VMWare installation. You can do this by logging in to your vCenter {webui}, and clicking btn:[Download trusted root CA certificates].

2. If the downloaded CA certificates file is in `.zip` format, extract the archive. The certificate files 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 provides you with some basic management features.

NOTE   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 {webui}, navigate to menu:Systems[Virtual Host Managers].

4. Click btn:[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 btn:[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 btn:[Refresh Data] to inventory the new VHM.

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": "11.11.12.13",
            "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-
427ffcddd88e",
                "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.

Troubleshooting Clients

Autoinstallation

Depending on your base channel, new autoinstallation profiles might be subscribed to a channel that is missing required packages.

For autoinstallation to work, these packages are required:

- `pyOpenSSL`
- `rhnlib`
- `libxml2-python`
- `spacewalk-koan`

To resolve this issue, check these things first:

- Check that the tools software channel related to the base channel in your autoinstallation profile is available to your organization and your user.
- Check that the tools channel is available to your Uyuni as a child channel.
- Check that the required packages and any dependencies are available in the associated channels.

Bare Metal Systems

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

- You must have the `pxe-default-image` package installed.
- File paths and parameters must be configured correctly. Check that the `vmlinuz0` and `initrd0.img` files, which are provided by `pxe-default-image`, are in the locations specified in the `rhn.conf` configuration file.
- Ensure the networking equipment connecting the bare metal system to the Uyuni server is working correctly, and that you can reach the Uyuni server IP address from the server.
- The bare metal system to be provisioned must have PXE booting enabled in the boot sequence, and must not be attempting to boot an operating system.
- The DHCP server must be responding to DHCP requests during boot. Check the PXE boot messages to ensure that:
  - the DHCP server is assigning the expected IP address
  - the DHCP server is assigning the the Uyuni server IP address as `next-server` for booting.
- Ensure Cobbler is running, and that the Discovery feature is enabled.

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

1. Select `pxe-default-profile` in the Cobbler menu with the arrow keys, and press the Tab key before the timer expires.

2. Add the kernel boot parameter `spacewalk-finally=running` using the integrated editor, and press Enter to continue booting.

3. Enter a shell with the username `root` and password `linux` to continue debugging.

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

Bootstrap End-of-Life {centos} 6 Clients

{centos} 6 is now at end-of-life, and the images provided in the client repository for this operating system are out of date. Bootstrapping new {centos} 6 clients using these packages will fail. This failure will not happen on {centos} 6 clients that are already installed and bootstrapped.

If you need to bootstrap new {centos} 6 clients you can edit the existing repositories to reflect the correct URL.

Procedure: Troubleshooting Bootstrapping a New {centos} 6 Client

1. On the {centos} 6 client, at the command prompt, open the `CentOS-Base.repo` file, located in the `/etc/yum.repos.d/` directory.

2. Locate the `mirrorlist` entry, pointing to `mirrorlist.centos.org`. There may be more than one entry. For example:

   ```
   mirrorlist=http://mirrorlist.centos.org/?release=6&arch=$basearch&repo=os
   ```

3. Comment out the `mirrorlist` entry, to prevent the package manager looking for the URL.

4. Edit the `baseurl` line to point to a `vault.centos.org` URL, and specify the {centos} 6 repositories. For example:

   ```
   baseurl=https://vault.centos.org/centos/6/os/$basearch/
   ```

5. Repeat for each repository listed in the file.

6. Bootstrap the client. For more information about bootstrapping {centos} clients, see **[Client-configuration › Clients-centos ›]**.

For more information about the {centos} 6 end-of-life, see http://mirror.centos.org/centos/6/readme.

Bootstrap Repository for End-of-Life Products

When supported products are synchronized, bootstrap repositories are automatically created and regenerated on the Uyuni Server. When a product reaches end-of-life and becomes unsupported, bootstrap repositories must be created manually if you want to continue using the product.

For more information about bootstrap repositories, see **[Client-configuration › Bootstrap-repository ›]**.

Procedure: Creating Bootstrap Repositories for End-Of-Life Products

1. At the command prompt on the Uyuni Server, as root, list the available unsupported bootstrap repositories with the `--force` option, for example:

   ```
   mgr-create-bootstrap-repo --list --force
   1. SLE-11-SP4-x86_64
   2. SLE-12-SP2-x86_64
   3. SLE-12-SP3-x86_64
   ```

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

   ```
   mgr-create-bootstrap-repo --create SLE-12-SP2-x86_64 --force
   ```

   If you do not want to create bootstrap repositories manually, you can check whether LTSS is available for the product and bootstrap repository you need.

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.

> **NOTE** 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 kbd:[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.

> **NOTE** 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 proxy appear in the {webui}, 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
```

Red Hat CDN Channel and Multiple Certificates

The {redhat} content delivery network (CDN) channels sometimes provide multiple certificates, but the Uyuni {webui} can only import a single certificate. If CDN presents a certificate that is different to the one the Uyuni {webui} knows about, validation fails and permission to access the repository is denied, even though the certificate is accurate. The error message received is:

```
[error]
Repository '<repo_name>' is invalid.
<repo.pem> Valid metadata not found at specified URL
History:
 - [|] Error trying to read from '<repo.pem>'
 - Permission to access '<repo.pem>' denied.
Please check if the URIs defined for this repository are pointing to a valid repository.
Skipping repository '<repo_nam' because of the above error.
Could not refresh the repositories because of errors.
HH:MM:SS RepoMDError: Cannot access repository. Maybe repository GPG keys are not imported
```

To resolve this issue, merge all valid certificates into a single `.pem` file, and rebuild the certificates for use by Uyuni:

Procedure: Resolving Multiple {redhat} CDN Certificates

1. On the {redhat} client, at the command prompt, as root, gather all current certificates from `/etc/pki/entitlement/` in a single `rh-cert.pem` file:

```
cat 866705146090697087.pem 3539668047766796506.pem redhat-entitlement-authority.pem >
rh-cert.pem
```

2. Gather all current keys from `/etc/pki/entitlement/` in a single `rh-key.pem` file:

```
cat 866705146090697087-key.pem 3539668047766796506-key.pem > rh-key.pem
```

You can now import the new certificates to the Uyuni Server, using the instructions in **[Client-configuration › Clients-rh-cdn ›]**.

Registration from {webui} fails and does not show any errors

For the initial registration from the {webui}, all Salt clients are using Salt SSH.

Because of its nature, Salt SSH clients do not report errors back to the server.

However, the Salt SSH clients store a log locally at `/var/log/salt-ssh.log` that can be inspected for errors.

Registering Older Clients

To register and use {centos} 6, {oracle} 6, {rhel} 6, or {sleses} 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 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 {webui}, you 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 needs 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.

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

GNU Free Documentation License

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

then there are none.

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

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

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

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

D. Preserve all the copyright notices of the Document.

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

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

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

H. Include an unaltered copy of this License.

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

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

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

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

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

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

O. Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

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