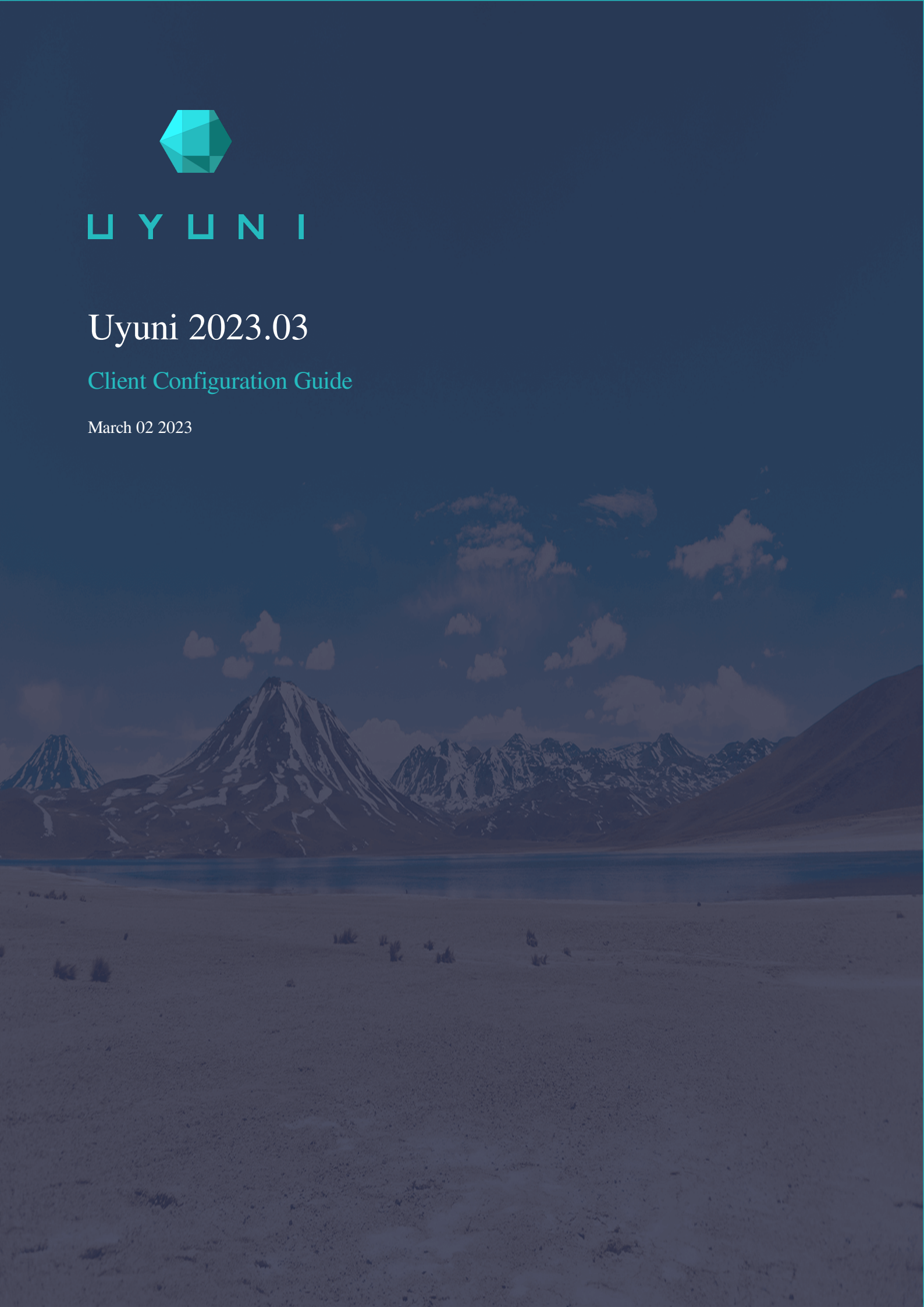


Uyuni 2023.03

Client Configuration Guide

March 02 2023

Table of Contents

Client Configuration Guide Overview

Updated: 2023-03-02

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 SUSE Linux Enterprise or another Linux operating system, with a range of hardware options.

For a complete list of supported clients and features, see **Client-configuration › Supported-features**.

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

Chapter 1. Supported Clients and Features

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

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

1.1. Supported Client Systems

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

The icons in this table indicate:

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

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

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

Table 1. Supported Client Systems

Operating System	Architecture	Traditional Clients	Salt Clients
SUSE Linux Enterprise 15	x86-64, ppc64le, IBM Z, ARM	✔	✔
SUSE Linux Enterprise 12	x86-64, ppc64le, IBM Z, ARM	✔	✔
SUSE Linux Enterprise Server for SAP 15	x86-64, ppc64le	✔	✔
SUSE Linux Enterprise Server for SAP 12	x86-64, ppc64le	✔	✔
SLE Micro	x86-64, ppc64le, aarch64	✖	✔
openSUSE Leap 15	x86-64, aarch64	✔	✔
Alibaba Cloud Linux 2	x86-64, aarch64	✖	✔

Operating System	Architecture	Traditional Clients	Salt Clients
AlmaLinux 9	x86-64, ppc64le, IBM Z, aarch64	✗	✓
AlmaLinux 8	x86-64, aarch64	✗	✓
Amazon Linux 2	x86-64, aarch64	✗	✓
CentOS 7	x86-64, ppc64le, aarch64	✓	✓
Debian 11	x86-64	✗	✓
Debian 10	x86-64	✗	✓
Oracle Linux 9	x86-64, aarch64	✗	✓
Oracle Linux 8	x86-64, aarch64	✗	✓
Oracle Linux 7	x86-64, aarch64	✓	✓
Red Hat Enterprise Linux 9	x86-64	✗	✓
Red Hat Enterprise Linux 8	x86-64	✗	✓
Red Hat Enterprise Linux 7	x86-64	✓	✓
Rocky Linux 9	x86-64, aarch64, ppc64le, s390x	✗	✓
Rocky Linux 8	x86-64, aarch64	✗	✓
Ubuntu 22.04	amd64	✗	✓
Ubuntu 20.04	amd64	✗	✓
Ubuntu 18.04	amd64	✗	✓

Debian and Ubuntu list the x86-64 architecture as amd64.

When the distibution reaches end-of-life, it enters grace period of 3 months when the support is considered deprecated. After that period, the product is considered unsupported. Any support may only be available on the best-effort basis.

For more information about end-of-life dates, see https://endoflife.software/operating-systems.

1.2. 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 SUSE Customer Center	✔
`spacewalk-hostname-rename`	Change the hostname of the Uyuni Server	✔
`spacewalk-clone-by-date`	Clone channels by a specific date	✔
`spacewalk-sync-setup`	Set up ISS master and slave organization mappings	✔
`spacewalk-manage-channel-lifecycle`	Manage channel lifecycles	✔

1.3. Supported SUSE and openSUSE Client Features

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

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

The icons in this table indicate:

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

Table 3. Supported Features on SUSE and openSUSE Operating Systems

Feature	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15
Client	✔	✔	✔
System packages	SUSE	SUSE	openSUSE Community
Registration	✔	✔	Salt
Install packages	✔	✔	Salt
Apply patches	✔	✔	Salt
Remote commands	✔	✔	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt
Organization custom states	Salt	Salt	Salt
System set manager (SSM)	✔	✔	Salt
Product migration	✔	✔	Salt
Basic Virtual Guest Management ✱	✔	✔	Salt
Advanced Virtual Guest Management ✱	Salt	Salt	Salt
Virtual Guest Installation (AutoYaST), as Host OS	Traditional	Traditional	✘
Virtual Guest Installation (image template), as Host OS	Salt	Salt	Salt
Virtual Guest Management	Salt	Salt	Salt
System deployment (PXE/AutoYaST)	✔	✔	✔
System redeployment (AutoYaST)	✔	✔	Salt

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

Feature	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15
Recurring Actions	Salt	Salt	Salt
AppStreams	N/A	N/A	N/A
Yomi	✗	✓	✓

✱ 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.

1.4. Supported SLE Micro Client Features

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

The icons in this table indicate:

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

Table 4. Supported Features on SLE Micro Operating Systems

Feature	SLE Micro
Client	Salt
Operating system packages	Salt
Registration	Salt
Install packages	Salt
Apply patches (requires CVE ID)	Salt

Feature	SLE Micro
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	Salt
Basic Virtual Guest Management ✱	Salt
Advanced Virtual Guest Management ✱	Salt
Virtual Guest Installation (Kickstart), as Host OS	Salt
Virtual Guest Installation (image template), as Host OS	Salt
System deployment (PXE/Kickstart)	Salt
System redeployment (Kickstart)	Salt
Contact methods	Salt: ZeroMQ
Works with Uyuni Proxy	Salt
Action chains	Salt
Staging (pre-download of packages)	?
Duplicate package reporting	Salt
CVE auditing (requires CVE ID)	Salt
SCAP auditing	?
Package verification	?
Package locking	Salt
System locking	?
Maintenance Windows	?
System snapshot	✗
Configuration file management	Salt

Feature	SLE Micro
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	Salt
Monitoring server	✗
Monitored clients	Salt
Docker buildhost	✗
Build Docker image with OS	✗
Kiwi buildhost	✗
Build Kiwi image with OS	Salt
Recurring Actions	Salt
AppStreams	N/A
Yomi	?

✱ 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.

1.5. openSUSE Leap Micro Client Features

openSUSE Leap Micro is supported by the SUSE community.

The icons in this table indicate:

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

Table 5. Supported Features on openSUSE Leap Micro Operating Systems

Feature	openSUSE Leap Micro
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	Salt
Basic Virtual Guest Management ✱	Salt
Advanced Virtual Guest Management ✱	Salt
Virtual Guest Installation (Kickstart), as Host OS	Salt
Virtual Guest Installation (image template), as Host OS	Salt
System deployment (PXE/Kickstart)	Salt
System redeployment (Kickstart)	Salt
Contact methods	Salt: ZeroMQ
Works with Uyuni Proxy	Salt
Action chains	Salt
Staging (pre-download of packages)	?
Duplicate package reporting	Salt
CVE auditing (requires CVE ID)	Salt
SCAP auditing	?
Package verification	?

Feature	openSUSE Leap Micro
Package locking	Salt
System locking	?
Maintenance Windows	?
System snapshot	✗
Configuration file management	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	Salt
Monitoring server	✗
Monitored clients	Salt
Docker buildhost	✗
Build Docker image with OS	✗
Kiwi buildhost	✗
Build Kiwi image with OS	Salt
Recurring Actions	Salt
AppStreams	N/A
Yomi	?

✱ 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.

1.6. Supported Alibaba Cloud Linux Features

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

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

The icons in this table indicate:

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

Table 6. Supported Features on Alibaba Cloud Linux Operating Systems

Feature	Alibaba Cloud Linux 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 ✱	?
Advanced Virtual Guest Management ✱	?
Virtual Guest Installation (Kickstart), as Host OS	✖
Virtual Guest Installation (image template), as Host OS	?
System deployment (PXE/Kickstart)	?

Feature	Alibaba Cloud Linux 2
System redeployment (Kickstart)	?
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	✗
Package locking	✗
System locking	✗
Maintenance Windows	✓
System snapshot	✗
Configuration file management	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	?
Monitoring server	✗
Monitored clients	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

✱ 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.

✱ The traditional stack is available on Alibaba Cloud Linux but it is unsupported.

1.7. Supported AlmaLinux Features

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

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:

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

Table 7. Supported Features on AlmaLinux Operating Systems

Feature	AlmaLinux 9	AlmaLinux 8
Client	Salt (plain AlmaLinux)	Salt (plain AlmaLinux)
System packages	AlmaLinux Community	AlmaLinux Community
Registration	Salt	Salt
Install packages	Salt	Salt
Apply patches	Salt	Salt
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	AlmaLinux 9	AlmaLinux 8
System set manager (SSM)	Salt	Salt
Product migration	N/A	N/A
Basic Virtual Guest Management ✱	Salt	Salt
Advanced Virtual Guest Management ✱	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✖	✖
Virtual Guest Installation (image template), as Host OS	Salt	Salt
System deployment (PXE/Kickstart)	Salt	Salt
System redeployment (Kickstart)	Salt	Salt
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	Salt	Salt
SCAP auditing	Salt	Salt
Package verification	✖	✖
Package locking	✖	✖
System locking	✖	✖
Maintenance Windows	✔	✔
System snapshot	✖	✖
Configuration file management	Salt	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	Salt	Salt

Feature	AlmaLinux 9	AlmaLinux 8
Monitoring server	✗	✗
Monitored clients	Salt	Salt
Docker buildhost	✗	✗
Build Docker image with OS	✗	✗
Kiwi buildhost	✗	✗
Build Kiwi image with OS	✗	✗
Recurring Actions	Salt	Salt
AppStreams	✓	✓
Yomi	N/A	N/A

✱ 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.

1.8. Supported Amazon Linux Features

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

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

The icons in this table indicate:

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

Table 8. Supported Features on Amazon Linux Operating Systems

Feature	Amazon Linux 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 ✱	?
Advanced Virtual Guest Management ✱	?
Virtual Guest Installation (Kickstart), as Host OS	✗
Virtual Guest Installation (image template), as Host OS	?
System deployment (PXE/Kickstart)	?
System redeployment (Kickstart)	?
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	✗

Feature	Amazon Linux 2
Package locking	✗
System locking	✗
Maintenance Windows	✓
System snapshot	✗
Configuration file management	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	?
Monitoring server	✗
Monitored clients	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

✱ 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.

✱ The traditional stack is available on Amazon Linux but it is unsupported.

1.9. Supported CentOS Features

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

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

The icons in this table indicate:

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

Table 9. Supported Features on CentOS Operating Systems

Feature	CentOS 7
Client	✔ (plain CentOS)
System packages	CentOS Community
Registration	✔
Install packages	✔
Apply patches (requires CVE ID)	✔ (third-party service required for errata)
Remote commands	✔
System package states	Salt
System custom states	Salt
Group custom states	Salt
Organization custom states	Salt
System set manager (SSM)	✔
Product migration	N/A
Basic Virtual Guest Management ✱	✔
Advanced Virtual Guest Management ✱	Salt
Virtual Guest Installation (Kickstart), as Host OS	Traditional
Virtual Guest Installation (image template), as Host OS	✔
System deployment (PXE/Kickstart)	✔
System redeployment (Kickstart)	✔

Feature	CentOS 7
Contact methods	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✔
Action chains	✔
Staging (pre-download of packages)	✔
Duplicate package reporting	✔
CVE auditing (requires CVE ID)	✔
SCAP auditing	✔
Package verification	Traditional
Package locking	✔
System locking	Traditional
Maintenance Windows	✔
System snapshot	Traditional
Configuration file management	✔
Snapshots and profiles	Traditional. Salt: Profiles supported, Sync not supported
Power management	✔
Monitoring server	✘
Monitored clients	Salt
Docker buildhost	✘
Build Docker image with OS	✘
Kiwi buildhost	✘
Build Kiwi image with OS	✘
Recurring Actions	Salt
AppStreams	N/A
Yomi	N/A

✱ 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.

1.10. Supported Debian Features

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

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

The icons in this table indicate:

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

Table 10. Supported Features on Debian Operating Systems

Feature	Debian 10	Debian 11
Client	✔	✔
System packages	Debian Community	Debian Community
Registration	Salt	Salt
Install packages	Salt	Salt
Apply patches	?	?
Remote commands	Salt	Salt
System package states	Salt	Salt
System custom states	Salt	Salt
Group custom states	Salt	Salt
Organization custom states	Salt	Salt
System set manager (SSM)	Salt	Salt
Product migration	N/A	N/A

Feature	Debian 10	Debian 11
Basic Virtual Guest Management ✱	Salt	Salt
Advanced Virtual Guest Management ✱	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✗	✗
Virtual Guest Installation (image template), as Host OS	Salt	Salt
System deployment (PXE/Kickstart)	✗	✗
System redeployment (Kickstart)	✗	✗
Contact methods	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	Salt	Salt
Action chains	Salt	Salt
Staging (pre-download of packages)	Salt	Salt
Duplicate package reporting	Salt	Salt
CVE auditing	?	?
SCAP auditing	?	?
Package verification	✗	✗
Package locking	✓	✓
System locking	✗	✗
Maintenance Windows	✓	✓
System snapshot	✗	✗
Configuration file management	Salt	Salt
Package profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓
Monitoring server	✗	✗
Monitoring clients	Salt	Salt

Feature	Debian 10	Debian 11
Docker buildhost	?	?
Build Docker image with OS	Salt	Salt
Kiwi buildhost	✗	✗
Build Kiwi image with OS	✗	✗
Recurring Actions	Salt	Salt
AppStreams	N/A	N/A
Yomi	N/A	N/A

✱ 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.

1.11. Supported Oracle Features

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

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

The icons in this table indicate:

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

Table 11. Supported Features on Oracle Linux Operating Systems

Feature	Oracle Linux 7	Oracle Linux 8	Oracle Linux 9
Client	✔	Salt	Salt

Feature	Oracle Linux 7	Oracle Linux 8	Oracle Linux 9
Operating system packages	✔	Salt	Salt
Registration	✔	Salt	Salt
Install packages	✔	Salt	Salt
Apply patches (requires CVE ID)	✔	Salt	Salt
Remote commands	✔	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
Product migration	N/A	N/A	N/A
Basic Virtual Guest Management ✱	✔	Salt	Salt
Advanced Virtual Guest Management ✱	Salt	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	Traditional	✘	✘
Virtual Guest Installation (image template), as Host OS	✔	Salt	Salt
System deployment (PXE/Kickstart)	✔	Salt	Salt
System redeployment (Kickstart)	✔	Salt	Salt
Contact methods	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Works with Uyuni Proxy	✔	Salt	Salt

Feature	Oracle Linux 7	Oracle Linux 8	Oracle Linux 9
Action chains	✔	Salt	Salt
Staging (pre-download of packages)	✔	Salt	Salt
Duplicate package reporting	✔	Salt	Salt
CVE auditing (requires CVE ID)	✔	Salt	Salt
SCAP auditing	✔	Salt	Salt
Package verification	Traditional	✖	✖
Package locking	✔	?	?
System locking	Traditional	✖	✖
Maintenance Windows	✔	✔	✔
System snapshot	Traditional	✖	✖
Configuration file management	✔	Salt	Salt
Snapshots and profiles	Traditional. Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✔	Salt	Salt
Monitoring server	✖	✖	✖
Monitored clients	Salt	Salt	Salt
Docker buildhost	✖	✖	✖
Build Docker image with OS	✖	✖	✖
Kiwi buildhost	✖	✖	✖
Build Kiwi image with OS	✖	✖	✖
Recurring Actions	Salt	Salt	Salt
AppStreams	N/A	✔	✔
Yomi	N/A	N/A	N/A

✱ 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.

1.12. Supported Red Hat Enterprise Linux Features

This table lists the availability of various features on native Red Hat Enterprise Linux clients.

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

The icons in this table indicate:

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

Table 12. Supported Features on Red Hat Enterprise Linux Operating Systems

Feature	RHEL 7	RHEL 8	RHEL 9
Client	✔	Salt	Salt
System packages	Red Hat	Red Hat	Red Hat
Registration	✔	Salt	Salt
Install packages	✔	Salt	Salt
Apply patches	✔	Salt	Salt
Remote commands	✔	Salt	Salt
System package states	Salt	Salt	Salt
System custom states	Salt	Salt	Salt
Group custom states	Salt	Salt	Salt

Feature	RHEL 7	RHEL 8	RHEL 9
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 ✱	✔	Salt	Salt
Advanced Virtual Guest Management ✱	Salt	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	Traditional	✘	✘
Virtual Guest Installation (image template), as Host OS	✔	Salt	Salt
System deployment (PXE/Kickstart)	✔	Salt	Salt
System redeployment (Kickstart)	✔	Salt	Salt
Contact methods	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	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	✔	Salt	Salt
SCAP auditing	✔	Salt	Salt
Package verification	Traditional	✘	✘
Package locking	✔	?	?
System locking	Traditional	✘	✘

Feature	RHEL 7	RHEL 8	RHEL 9
Maintenance Windows	✔	✔	✔
System snapshot	Traditional	✖	✖
Configuration file management	✔	Salt	Salt
Snapshots and profiles	Traditional. Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✔	Salt	Salt
Monitoring server	✖	✖	✖
Monitored clients	Salt	Salt	Salt
Docker buildhost	✖	✖	✖
Build Docker image with OS	?	?	?
Kiwi buildhost	✖	✖	✖
Build Kiwi image with OS	✖	✖	✖
Recurring Actions	Salt	Salt	Salt
AppStreams	N/A	✔	✔
Yomi	N/A	N/A	N/A

✱ 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.

1.13. Supported Rocky Linux Features

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

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

The icons in this table indicate:

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

Table 13. Supported Features on Rocky Linux Operating Systems

Feature	Rocky Linux 8	Rocky Linux 9
Client	Salt (plain Rocky Linux)	Salt (plain Rocky Linux)
System packages	Rocky Linux Community	Rocky Linux Community
Registration	Salt	Salt
Install packages	Salt	Salt
Apply patches	Salt	Salt
Remote commands	Salt	Salt
System package states	Salt	Salt
System custom states	Salt	Salt
Group custom states	Salt	Salt
Organization custom states	Salt	Salt
System set manager (SSM)	Salt	Salt
Product migration	N/A	N/A
Basic Virtual Guest Management ✱	Salt	Salt
Advanced Virtual Guest Management ✱	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✖	✖

Feature	Rocky Linux 8	Rocky Linux 9
Virtual Guest Installation (image template), as Host OS	Salt	Salt
System deployment (PXE/Kickstart)	Salt	Salt
System redeployment (Kickstart)	Salt	Salt
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	Salt	Salt
SCAP auditing	Salt	Salt
Package verification	✗	✗
Package locking	?	?
System locking	✗	✗
Maintenance Windows	✓	✓
System snapshot	✗	✗
Configuration file management	Salt	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	Salt	Salt
Monitoring server	✗	✗
Monitored clients	Salt	Salt
Docker buildhost	✗	✗
Build Docker image with OS	✗	✗
Kiwi buildhost	✗	✗
Build Kiwi image with OS	✗	✗
Recurring Actions	Salt	Salt

Feature	Rocky Linux 8	Rocky Linux 9
AppStreams	✔	✔
Yomi	N/A	N/A

✱ 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.

1.14. Supported Ubuntu Features

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

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

The icons in this table indicate:

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

Table 14. Supported Features on Ubuntu Operating Systems

Feature	Ubuntu 18.04	Ubuntu 20.04	Ubuntu 22.04
Client	✔	✔	✔
System packages	Canonical	Canonical	Canonical
Registration	Salt	Salt	Salt
Install packages	Salt	Salt	Salt
Apply patches	✔	✔	✔
Remote commands	Salt	Salt	Salt

Feature	Ubuntu 18.04	Ubuntu 20.04	Ubuntu 22.04
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
Product migration	N/A	N/A	N/A
Basic Virtual Guest Management ✱	Salt	Salt	Salt
Advanced Virtual Guest Management ✱	Salt	Salt	Salt
Virtual Guest Installation (Kickstart), as Host OS	✖	✖	✖
Virtual Guest Installation (image template), as Host OS	Salt	Salt	Salt
System deployment (PXE/Kickstart)	✖	✖	✖
System redeployment (Kickstart)	✖	✖	✖
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	?	?	?
SCAP auditing	?	?	?
Package verification	✖	✖	✖
Package locking	✔	✔	✔

Feature	Ubuntu 18.04	Ubuntu 20.04	Ubuntu 22.04
System locking	✗	✗	✗
System snapshot	✗	✗	✗
Configuration file management	Salt	Salt	Salt
Package profiles	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported	Salt: Profiles supported, Sync not supported
Power management	✓	✓	✓
Monitoring	Salt	Salt	Salt
Docker buildhost	?	?	?
Build Docker image with OS	Salt	Salt	Salt
Kiwi buildhost	✗	✗	✗
Build Kiwi image with OS	✗	✗	✗
Recurring Actions	Salt	Salt	Salt
AppStreams	N/A	N/A	N/A
Yomi	N/A	N/A	N/A

✱ 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.

Chapter 2. Configuration Basics

Uyuni requires a number of steps to prepare the environment for clients registration before a wide range of its operations can be utilized.

This section contains summary of the initial configuration steps that are necessary to support environment operations following successful Uyuni installation and setting up.

- For more information about installing Uyuni, see **Installation-and-upgrade › Install-uyuni**.
- For more information about setting up Uyuni, see **Installation-and-upgrade › Uyuni-server-setup**.

2.1. Software Channels

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

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

A base channel consists of packages built for a specific operating system type, version, and architecture. For example, the SUSE Linux Enterprise Server 15 x86-64 base channel contains only software compatible with that operating system and architecture.

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

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

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

2.1.1. Packages Provided by SUSE Package Hub

SUSE Package Hub is an extension to SUSE Linux Enterprise products that provides additional open source software provided by the openSUSE community.

The packages in SUSE Package Hub are provided by the openSUSE community. They are not supported by SUSE.

If you are using SUSE Linux Enterprise operating systems on your clients, you can enable the SUSE Package Hub extension to access these additional packages. This provides the SUSE Package Hub channels, which you can subscribe your clients to.

SUSE Package Hub 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 SUSE Package Hub 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 SUSE Package Hub packages. You can then explicitly enable the specific packages you require. For more information about content lifecycle management, see **Administration › Content-lifecycle**.

2.1.2. Packages Provided by AppStream

For Red Hat 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 Web UI, 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**.

2.1.3. Packages Provided by EPEL

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

The packages in EPEL are provided by the Fedora community. They are not supported by SUSE.

If you are using Red Hat 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**.

2.1.4. Unified Installer Updates Channels on SUSE Linux Enterprise Clients

This channel is used by the Unified Installer to ensure it is up to date before it installs the operating system. All SUSE Linux Enterprise products should have access to the installer updates channel during installation.

For SUSE Linux Enterprise Server 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 SUSE Linux Enterprise variants, including SUSE Linux Enterprise for SAP, you must add the installer updates channel manually. To do this, clone the appropriate SUSE Linux Enterprise Server installer updates channel below the base channel of these SUSE Linux Enterprise variants. When creating an autoinstallable distribution for these SUSE Linux Enterprise variants after the channel was cloned, it is used automatically.

2.1.5. 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**.

2.1.5.1. Local Repository Locations

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

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.

For Salt clients, local repositories will be disabled each time a channel state is executed. For example, when applying the highstate or performing a package action.

If local repositories should stay enabled after onboarding the following pillar must be set for the affected Salt client:

Edit the `/srv/pillar/top.sls` file:

```
base:
  'minionid':
    - localrepos
```

Edit the `/srv/pillar/localrepos.sls` file:

```
mgr_disable_local_repos: False
```

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
SUSE Linux Enterprise Server	`/etc/zypp/repos.d`
openSUSE	`/etc/zypp/repos.d`
Red Hat Enterprise Linux	`/etc/yum.repos.d/`
CentOS	`/etc/yum.repos.d/`
Ubuntu	`/etc/apt/sources.list.d/`
Debian	`/etc/apt/sources.list.d/`

2.1.6. Software Products

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

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

Procedure: Adding Software Channels

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

For more information, see **Installation-and-upgrade › Setup-wizard**.

2.2. 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.

2.2.1. 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 from the Web UI

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

Procedure: Checking Synchronization Progress from the Command Prompt

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

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

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

2.2.2. Options for Automatic Mode

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

Flush Mode::

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::

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
```

2.2.2.1. 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
```

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.

2.2.3. Manually Generate a Bootstrap Repository

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

Procedure: Generating the Bootstrap Repository for SUSE Linux Enterprise

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

   ```
   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
   ```

3. Alternatively, use the number shown next to the distribiution name in the list of available distributions.

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
```

2.2.3.1. Repositories with Multiple Architectures

If you are creating bootstrap repositories that include multiple different architectures, you need to be careful that all architectures are updated correctly. For example, the x86-64 and IBM Z architectures for SLE use the same bootstrap repository URL at `/srv/www/htdocs/pub/repositories/sle/15/2/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
```

2.2.4. 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**.

2.3. Activation Keys

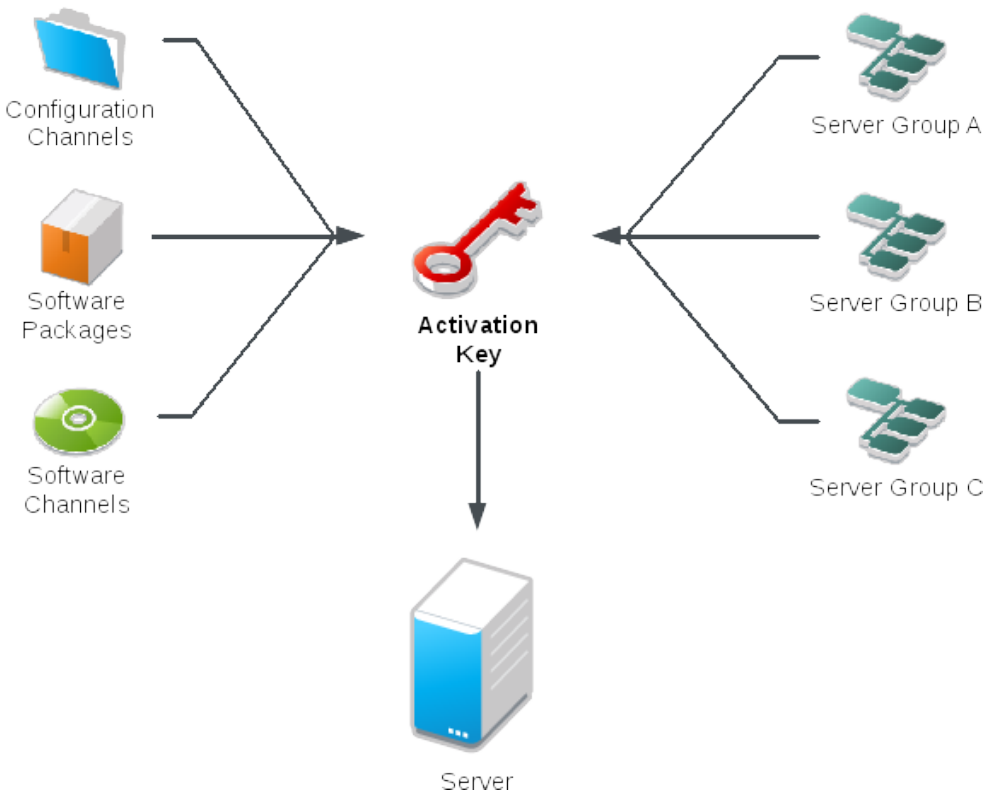
Activation keys are used with traditional and Salt clients to ensure that your clients have the correct software entitlements, are connecting to the appropriate channels, and are subscribed to the relevant groups. Each activation key is bound to an organization, which you can set when you create the 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 Web UI, as an administrator, navigate to **Systems › Activation Keys**.
2. Click the **[Create Key]** button.
3. On the Activation Key Details page, in the Description field, enter a description of the activation key.
4. In the Key field, enter a name for the activation key. For example, SLES15-SP4 for SUSE Linux Enterprise Server 15 SP4.

Do not use commas in the Key field for any SUSE products. However, you **must** use commas for Red Hat Products.

5. In the Base Channels drop-down box, select the appropriate base software channel, and allow the relevant child channels to populate. For more information, see reference:admin/setup-wizard.pdf and **Administration › Custom-channels**.
6. Select the child channels you need (for example, the mandatory SUSE Manager tools and updates channels).
7. 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 **[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 **[Update Activation Key]** to save this change.

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

2.3.1. 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.

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 **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.

2.3.2. 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 **Details** › **Reactivation** tab. Click **`[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 **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
```

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.

2.3.3. 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:

```
sles15-sp4-web_server-room_129-x86_64
```

```
sles15-sp4-test_packages-blg_502-room_21-ppc64le
```

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

Included Channels

When creating activation keys you also need to keep in mind which software channels 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**.

2.4. 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.

RPM packages can be signed directly, while Debian based systems sign only the metadata and use a chain of checksums to secure the packages. Most RPM based systems use signed metadata in addition to signed packages.

2.4.1. Trust GPG Keys on Clients

Operating systems either trust their own GPG keys directly or at least ship them installed with the minimal system. But third party packages signed by a different GPG key need manual handling. 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 use now GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key gets trusted as soon as

the channel is refreshed or the first package gets installed from this channel.

The GPG key URL which is set of a software channel must exist. In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs.