Uyuni 2026.11

Retail Guide



UYUNI

Chapter 1. Preface

Retail
Uyuni 2026.11

Uyuni for Retail 2026.11 is an **open-source infrastructure management solution**, specifically **optimized for the retail industry**. Built on the same technology as Uyuni, it has been tailored to meet the unique operational needs of retail organizations.

Designed for **retail environments**, Uyuni for Retail supports **point-of-service (POS) terminals**, allowing businesses to manage transactions, promotions, and customer loyalty programs. Beyond retail, it can also be used in **educational institutions** to maintain student computers, as well as in **banks and hospitals** for self-service kiosks.

Uyuni for Retail is suited for deployments that include **servers, workstations, point-of-service terminals, and other connected devices**. It enables administrators to **install, configure, and update software across their infrastructure**, while streamlining **deployment and provisioning** of POS machines.

This guide provides a **comprehensive overview** of Uyuni for Retail, covering its **initial installation and setup**.

For additional guidance, refer to the **Uyuni documentation suite** at https://www.uyuni-project.org/uyuni-docs.

For more details on managing your Uyuni for Retail environment or seeking assistance, see **Retail Guide › Retail Next**.

Publication Date: 2026-11-25

Table of Contents

1. Preface 1
2. Components 5
   2.1. The Uyuni Server 5
   2.2. Build Hosts 5
   2.3. Branch Servers 5
   2.4. Point-of-Service Terminals 5
   2.5. Fitting It All Together 6
      2.5.1. Hardware Types 6
      2.5.2. Branch System Groups 6
      2.5.3. Salt Formulas 6
      2.5.4. Cobbler 7
      2.5.5. Saltboot 7
3. Retail Requirements 8
   3.1. Server Requirements 8
   3.2. Branch Server Requirements 9
   3.3. Build Host Requirements 9
   3.4. Network Requirements 10
   3.5. POS Terminal Requirements 10
4. Network Architecture 11
   4.1. Branch Server Network Configuration 11
      4.1.1. Shared Network Architecture 11
      4.1.2. Externally Managed Dedicated Network Architecture 12
      4.1.3. Dedicated Network Architecture 12
5. Retail Installation 14
   5.1. Install Uyuni Retail Server with openSUSE 14
      5.1.1. Install Uyuni on openSUSE Leap 14
   5.2. Retail Uyuni Server Setup 15
      5.2.1. Set up Uyuni with YaST 15
      5.2.2. Create the Main Administration Account 16
   5.3. Retail Uyuni Branch Server 16
      5.3.1. Add Software Channels 17
      5.3.2. Check Synchronization Status 18
      5.3.3. Trust GPG Keys on Clients 18
      5.3.4. Create Activation Key for a Branch Server and the Retail Terminal Images 19
      5.3.5. Register the Branch Server and Terminals as Clients 19
   5.4. Set Up the Uyuni for Retail Environment 20
      5.4.1. Prepare and Build Terminal Images 21
      5.4.2. Branch Identification 21
      5.4.3. Required System Groups 22
      5.4.4. Configure Network Services for Saltboot 23
      5.4.5. Terminal Partitioning and Image Selection 24
      5.4.6. Synchronize Images to the Branch Server 24
6. Deploying Terminals 25

6.1. Deployment basics 25
6.1.1. Create A Hardware Type Group 25
6.1.2. Assign and Configure the Saltboot Formula for Each Hardware Type Group 26
6.1.3. Synchronize Images to the Branch Server 26
6.1.4. Deploy Images to the Terminals 27
6.1.5. Customize the Terminal Image Download Process 27
6.2. Deploy Terminals - Other Methods 28
6.2.1. Deploy Terminals with a Removable USB Device 28
6.2.2. Deploy Terminals over a Wireless Network 29
6.3. Deploy Terminals and Auto-Accept Keys 32
6.3.1. Configure the server to auto-accept 33
6.3.2. Configure Saltboot to keep auto-signed grains 34
6.3.3. Configure Saltboot to auto-sign during PXE boot 35
6.4. Forced Saltboot image redeployment 35
6.4.1. Force Saltboot redeployment using Salt grains 36
6.4.2. Force Saltboot redeployment using custom info values 36
6.4.3. Force Saltboot redeployment using Saltboot API call 38
6.4.4. Force Saltboot redeployment by custom pillar 38
6.5. Terminal Boot Process (Saltboot Diagram) 39
6.6. Terminal Names 41
6.6.1. Naming by HWType 41
6.6.2. Naming by Hostname 41
6.6.3. Naming by FQDN 42
6.6.4. Naming by MAC 42
6.6.5. Assign Hostnames to Terminals 43
6.7. Offline Use 44
6.7.1. Offline Terminal Reboot 44
6.7.2. Cached Terminal Updates 45
6.8. Rate Limiting Terminals 45
6.8.1. Troubleshooting 45
7. Introduction to Retail Formulas 46
7.1. Branch server formulas 46
7.1.1. Containerized branch server (Uyuni 2026.11) 46
7.2. Partitioning and image deployment formula 47
8. Image Pillars 48
9. Administration 50
9.1. Mass Configuration 50
9.1.1. Branch Server Mass Configuration 50
9.1.2. Terminal Mass Configuration 51
9.1.3. Export Configuration to Mass Configuration File 52
9.2. Example YAML File for Mass Configuration 52
9.3. Delta Images 54
9.3.1. Building Delta Images 54
9.3.2. Tuning Delta Generation 55
9.3.3. Image Synchronizing to the Branch Server 55
10. Upgrade Uyuni for Retail Branch Server 56
11. Example configurations 57

11.1. Configurations Using Uyuni for Retail Branch Proxy 5.0 and Later 57
11.2. Configurations Using Uyuni for Retail Branch Proxy 4.3 57
11.3. Set Up the Uyuni for Retail Environment using containerized proxy 57
11.3.1. Requirements and assumptions 58
11.3.2. Create required system groups 58
11.3.3. Saltboot group 58
11.3.4. Comparing containerized and non-containerized workflows 59
11.3.5. Validating Saltboot group configuration 60
12. Best practices 63
12.1. Deploying new images 63
12.1.1. Controlling image versions 63
13. What Next? 65
13.1. More Documentation 65
13.2. Support 65
14. GNU Free Documentation License 66

Chapter 2. Components

Uyuni for Retail is made up of various components. For more on how these components work together, see **Retail Guide › Retail Network Arch**.

2.1. The Uyuni Server

The Uyuni server contains information about infrastructure, network topology, and everything required to automate image deployment and perform day-to-day operations on branches and terminals. This can include database entries of registered systems, Salt pillar data for images, image assignments, partitioning, network setup, network services, and more.

2.2. Build Hosts

Build hosts can be arbitrary servers or virtual machines. They are used to securely build operating system images.

For more information on build hosts, see **Administration Guide › Image Management**.

2.3. Branch Servers

Branch servers should be physically located close to point-of-service terminals, such as in an individual store or branch office. Branch servers provide services for PXE boot, and act as an image cache, Salt broker, and proxy for software components (RPM packages). The branch server can also manage local networking, and provide DHCP and DNS services.



- For monitoring, you can install Prometheus server on the Branch servers.
- For visualization and analysis, you can install Grafana with multiple data sources on a dedicated host. For more information about Prometheus and Grafana, see **Administration Guide › Monitoring**.

2.4. Point-of-Service Terminals

Point-of-Service (POS) terminals can come in many different formats, such as point-of-sale terminals, kiosks, digital scales, self-service systems, and reverse-vending systems. Every terminal, however, is provided by a vendor, who set basic information about the device in the firmware. Uyuni for Retail accesses this vendor information to determine how best to work with the terminal in use.

In most cases, different terminals will require a different operating system (OS) image to ensure they work correctly. For example, an information kiosk has a high-resolution touchscreen, where a cashier terminal might only have a very basic display. While both of these terminals require similar processing and network

functionality, they will require different OS images. The OS images ensure that the different display mechanisms work correctly.

Uyuni for Retail supports POS terminals that boot using both BIOS and UEFI. For UEFI booting terminals, you will need to configure the EFI partition in the Saltboot formula. For more information on EFI in the Saltboot formula, see **Specialized Guides › Salt › Salt Formula Saltboot**.

2.5. Fitting It All Together

Uyuni for Retail uses the same technology as Uyuni, but is customized to address the needs of retail organizations.

2.5.1. Hardware Types

Because every environment is different, and can contain many different configurations of many different terminals, Uyuni for Retail uses hardware types to simplify device management.

Hardware types allow you to group devices according to manufacturer and device name. Then all devices of a particular type can be managed as one.

2.5.2. Branch System Groups

Uyuni for Retail uses system groups to organize the various devices in your environment.

Each branch requires a system group, containing a single branch server, and the POS terminals associated with that server. Each system group is identified with a Branch ID. The Branch ID is used in formulas and scripts to automatically update the entire group.

2.5.3. Salt Formulas

Uyuni for Retail uses Salt formulas to help simplify configuration. Formulas are pre-written Salt states, that are used to configure your installation.

You can use formulas to apply configuration patterns to your hardware groups. Uyuni for Retail uses the Saltboot formula, which defines partitioning and OS images for terminals.

You can use default settings for formulas, or edit them to make them more specific to your environment.

For more information about formulas, see **Retail Guide › Retail Formulas Intro**.

2.5.4. Cobbler

Uyuni for Retail uses Cobbler to manage and prepare boot configuration for the POS terminals.

2.5.5. Saltboot

Saltboot is a collection of tools and processes that are used to bootstrap, deploy and validate Uyuni for Retail terminals.

Saltboot consists of:

- Initialization:

  The Saltboot initrd is created during image building and is required for bootstrapping terminals.

- Saltboot state:

  The Salt state that contains the logic for the entire Saltboot process.

- Partitioning pillars:

  The Salt pillar structure that describes how terminals are partitioned and what image is deployed on each terminal.

- Images and boot images pillars:

  When the image building feature in Uyuni successfully builds an image that contains the Saltboot initrd, the image and boot image Salt pillars are created.

The Saltboot process involves the Uyuni Server, a terminal running the Saltboot initrd, and the branch server providing the Saltboot services to the terminal.

For a detailed diagram explaining how the Saltboot boot process works, see **Retail Guide › Retail Saltboot Diagram**.

Chapter 3. Retail Requirements

Before you install Uyuni for Retail, ensure your environment meets the minimum requirements. This section lists the requirements for the Uyuni for Retail installation.

Review also general requirements of the Uyuni listed at **Installation and Upgrade Guide › General Requirements**.



Uyuni for Retail is tested on x86-64 architecture.

3.1. Server Requirements

Table 1. Server Hardware Requirements

Hardware	Details	Recommendation
CPU	x86-64, aarch64, ppc64le, or s390x	Minimum 4 dedicated 64-bit CPU cores
RAM	Minimum	16 GB
	Recommended	32 GB
Disk Space	/ (root directory)	40 GB
	/var/lib/containers/storage/volumes	Minimum 150 GB (depending on the number and type of the product*)
	/var/lib/containers/storage/volumes/var-pgsql	Minimum 50 GB
Swap space	Systems can benefit from additional swap space. SUSE recommends using a swap file instead of a swap partition. For more information about swap space, see **Installation and Upgrade Guide › Swap space**.	8 to 12 GB



Uyuni performance depends on hardware resources, network bandwidth, latency between clients and server, etc.

Based on the experience and different deployments that are in use, the advice for optimal performance of Uyuni Server with an adequate number of proxies is to not exceed 10,000 clients per single server. It is highly recommended to move to the Hub setup and involve consultancy when you have more than 10,000 clients. Even with fine-tuning and an adequate number of proxies, such a large number of clients can lead to performance issues.

> For more information about managing a large number of clients, see **Specialized Guides › Large Deployments › Hub Multi Server**.

3.2. Branch Server Requirements

Table 2. Branch Server Hardware Requirements

Hardware	Details	Recommendation
CPU	x86-64	Minimum 2 dedicated 64-bit CPU cores
	Recommended	The same as minimum values
RAM	Minimum	2 GB
	Recommended	8 GB
Disk Space	/ (root directory)	Minimum 40 GB
	/var/lib/containers/storage/volumes	Minimum 100 GB
Swap space	Systems can benefit from additional swap space. SUSE recommends using a swap file instead of a swap partition. For more information about swap space, see **Installation and Upgrade Guide › Hardware Requirements › Installation Swap Space**.	4 to 8 GB

3.3. Build Host Requirements

Table 3. Hardware Requirements for Build Host

Hardware	Recommended
CPU	Multi-core 64-bit CPU
RAM:	Test Server Minimum 2 GB
	Production Server Minimum 4 GB
Disk Space:	/ (root) Minimum 24 GB
	/var/lib/Kiwi Minimum 15 GB

3.4. Network Requirements

- The Uyuni Server requires a reliable and fast WAN connection.
- The branch server requires a reliable WAN connection, to reach the Uyuni Server.
- If you are using a dedicated network, the branch server requires at least two network interfaces: one connected to the WAN with a reachable Uyuni Server, and one connected to the internal branch LAN.
- Terminals require a LAN connection to the branch server network.

3.5. POS Terminal Requirements

Table 4. Hardware Requirements for Terminals

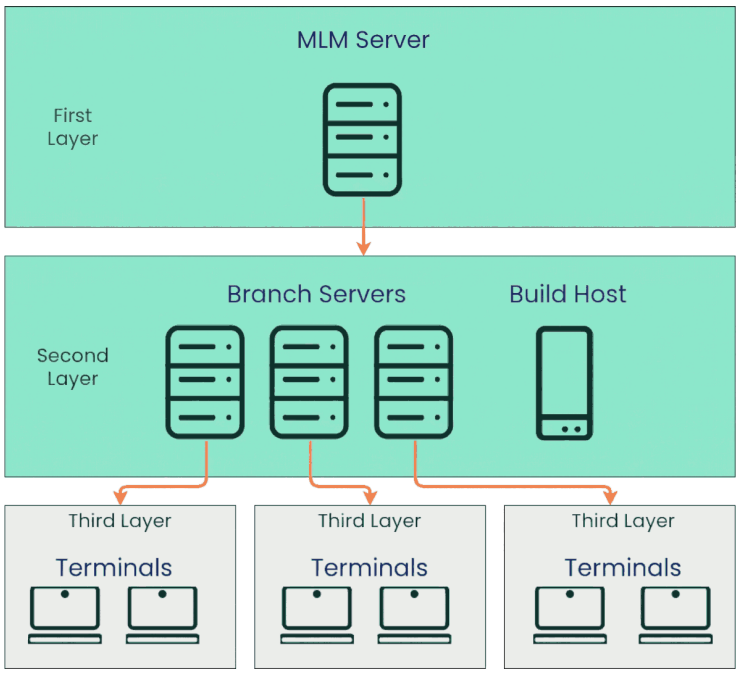
Hardware	Recommended
RAM:	Minimum 1 GB for hosts that need to run OS images built with Kiwi (PXE booted or not)
Disk Space:	Disk space depends on size of the OS image

For more information on Uyuni for Retail POS terminals, see documentation on Uyuni Salt clients (**Client Configuration Guide › Uyuni Client Config Overview**).

Chapter 4. Network Architecture

Uyuni for Retail uses a layered architecture:

- The first layer contains the Uyuni Server.
- The second layer contains one or more branch servers to provide local network and boot services. It also contains one or more build hosts.
- The final layer contains any number of deployed point-of-service terminals.



Branch servers should be physically located close to point-of-service terminals, such as in an individual store or branch office. We recommend you have a fast network connection between the branch server and its terminals. Branch servers provide services for PXE boot, and act as an image cache, Salt broker, and proxy for software components (RPM packages). The branch servers can also manage local networking, and provide DHCP and DNS services.

Uyuni for Retail Branch Servers are implemented as enhanced Uyuni Proxies. For technical background information on Uyuni Proxies, see **Installation and Upgrade Guide › Container Deployment › Proxy Container Setup Uyuni**.

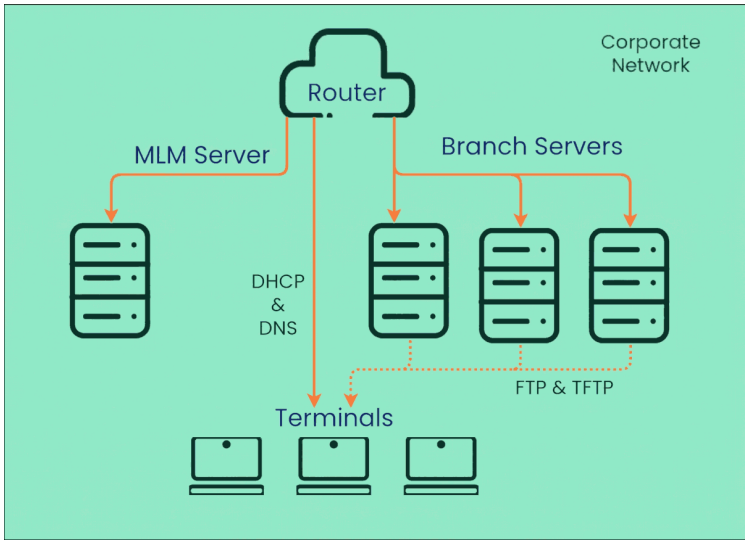
4.1. Branch Server Network Configuration

You can use branch servers in different network configurations, depending on your installation requirements.

4.1.1. Shared Network Architecture

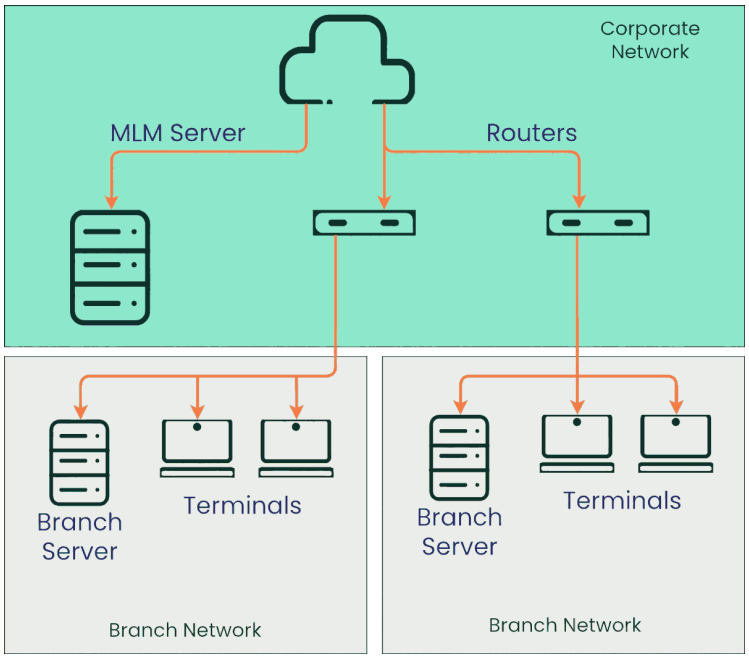
The branch server and the terminals are connected to the same network as the Uyuni Server. In this

configuration, external routers provide DHCP and DNS services to the branch servers and the terminals, and the branch server provides PXE, HTTP and TFTP services to the terminals in their branch network.



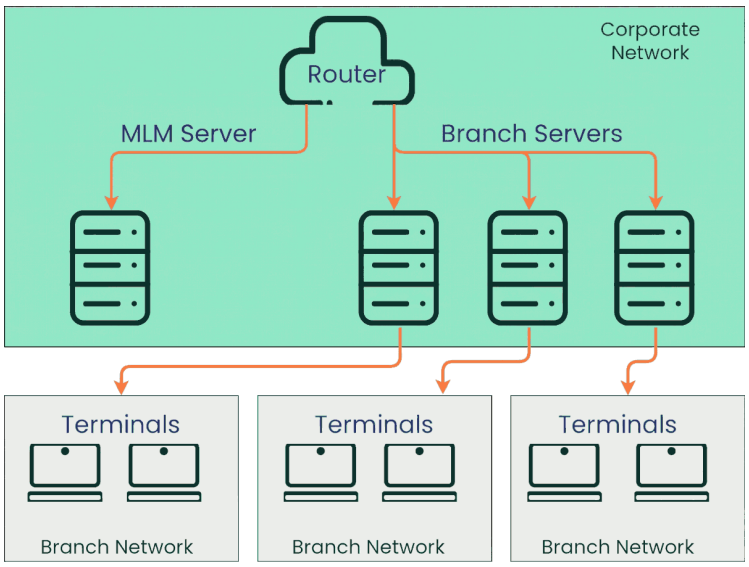
4.1.2. Externally Managed Dedicated Network Architecture

The branch servers are in separate branch networks, along with the terminals they manage. In this configuration, external routers provide DHCP and DNS services to the branch servers and the terminals, and the branch server provides proxy, PXE, HTTP and TFTP services to the terminals in their branch network.



4.1.3. Dedicated Network Architecture

The branch servers are in the same network as the Uyuni Server, and terminals use an isolated branch network. In this configuration, the branch servers are in the corporate network, and provide all DHCP, DNS, PXE, HTTP and TFTP services to the terminals in the branch networks.


Corporate
Network
Router
MLM Server
Branch Servers
Terminals
Branch Network
Terminals
Branch Network
Terminals
Branch Network

Chapter 5. Retail Installation

Uyuni Retail Server and Uyuni Retail Branch Server are already part of Uyuni Server and Proxy containers.

5.1. Install Uyuni Retail Server with openSUSE

Uyuni for Retail Server can be installed on openSUSE.

For general requirements, see **Installation and Upgrade Guide › Uyuni Install Requirements**.



For more information about the latest version and updates of openSUSE Leap, see https://doc.opensuse.org/release-notes/.

5.1.1. Install Uyuni on openSUSE Leap

You install Uyuni as an add-on to openSUSE Leap.

Procedure: Installing Uyuni on openSUSE Leap

1. As the base system, install openSUSE Leap with all available service packs and package updates applied.
2. Configure a resolvable fully qualified domain name (FQDN) with **yast › System › Network Settings › Hostname/DNS**.
3. Set variables to use to create repository:

```
repo=repositories/systemsmanagement:/
repo=${repo}Uyuni:/Stable/images/repo/Uyuni-Server-POOL-x86_64-Media1/
```

4. Add the repository for installing the Uyuni Server software as root:

```
zypper ar https://download.opensuse.org/$repo uyuni-server-stable
```

5. Refresh metadata from the repositories as root, and confirm the import of new GPG key into the keyring:

```
zypper ref
```

6. Install the pattern for the Uyuni Server as root:

```
zypper in patterns-uyuni_server
```

7. Install the pattern for the Uyuni for Retail product as root:

```
zypper in patterns-uyuni_retail
```

8. Reboot the Uyuni for Retail Server.

Continue with the server setup as described in **Retail Guide › Retail Uyuni Server Setup**.

5.2. Retail Uyuni Server Setup

This section covers Uyuni for Retail Server setup, using these procedures:

- Set up Uyuni with YaST
- Create the main administration account
- Add Software Channels
- Check Synchronization Status
- Trust GPG Keys on Clients
- Register the Branch Server and Terminals as Clients

5.2.1. Set up Uyuni with YaST

This section guides you through Uyuni setup procedures.

Procedure: Uyuni Setup

1. On the Uyuni Server, at the command prompt, as root, start YaST:

```
yast2
```

2. Navigate to **Network Services › Uyuni Setup** to begin set up.
3. From the introduction screen, select **Uyuni Setup › Set up Uyuni from scratch** and click **[Next]** to continue.
4. Type an email address to receive status notifications and click **[Next]** to continue. Uyuni can sometimes send a large volume of notification emails. You can disable email notifications in the Web UI after setup, if you need to.
5. Type your certificate information and provide a password. Passwords must be at least seven characters in length, and must not contain spaces, single or double quotation marks (' or "), exclamation marks (!), or dollar signs ($). Always store your passwords in a secure location. You must have the certificate password to set up the Uyuni Proxy.

```
Click btn:[Next] to continue.
```

6. Navigate to **Uyuni Setup › Database Settings** screen, type a database username and password, and click

[Next] to continue. Passwords must be at least seven characters in length, and must not contain spaces, single or double quotation marks (' or "), exclamation marks (!), or dollar signs ($). Always store your passwords in a secure location.

```
Click btn:[Next] to continue.
```

7. Click **[Yes]** to begin the setup process.
8. When setup is complete, click **[Next]** to continue. Take note of the address to access the Uyuni Web UI.
9. Click **[Finish]** to complete Uyuni setup.

5.2.2. Create the Main Administration Account

This section covers how to create your organization's main administration account for Uyuni.



The main administration account has the highest authority within Uyuni. Ensure you keep access information for this account secure. We recommend that you create lower level administration accounts for organizations and groups. Do not share the main administration access details.

Procedure: Setting Up the Main Administration Account

1. In your web browser, enter the address for the Uyuni Web UI. This address was provided after you completed setup. For more information, see **Retail Guide › Retail Uyuni Server Setup › Retail Server Setup Yast**.
2. Sign in to the Web UI, navigate to the **Create Organization › Organization Name** field, and enter your organization name.
3. In the **Create Organization › Desired Login** and **Create Organization › Desired Password** fields, enter your username and password.
4. Complete the Account Information fields, including an email for system notifications.
5. Click **[Create Organization]** to finish creating your administration account.

When you have completed the Uyuni Web UI setup, you are taken to the **Home › Overview** page.

5.3. Retail Uyuni Branch Server

This section covers Uyuni for Retail Branch Server installation and setup, using these procedures:

- Add Software Channels
- Check Synchronization Status

- Trust GPG Keys on Clients
- Register the Branch Server and Terminals as Clients

The Uyuni for Retail Branch Server is a Uyuni Proxy with additional Retail features. For proxy installation procedures, see **Installation and Upgrade Guide › Container Deployment › Proxy Container Setup Uyuni**.

Then continue with the following sections.

5.3.1. Add Software Channels

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

The products you need for this procedure are:

Table 5. openSUSE Channels - CLI

OS Version	openSUSE Leap 15.4
Base Channel	opensuse_leap15_4
Client Channel	opensuse_leap15_4-uyuni-client
Updates Channel	opensuse_leap15_4-updates
Non-OSS Channel	opensuse_leap15_4-non-oss
Non-OSS Updates Channel	opensuse_leap15_4-non-oss-updates
Backports Updates Channel	opensuse_leap15_4-backports-updates
SLE Updates Channel	opensuse_leap15_4-sle-updates

Procedure: Adding Software Channels at the Command Prompt

1. With the spacewalk-common-channels command you can add the appropriate channels. At the command prompt on the Uyuni container host, as root, execute the following command:

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

2. Synchronize the channels:

```
mgrctl exec -- mgr-sync refresh --refresh-channels
```

5.3.2. Check Synchronization Status

Procedure: Checking Synchronization Progress

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

Procedure: Checking Synchronization Progress from the Command Prompt

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

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

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



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

5.3.3. Trust GPG Keys on Clients

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

Procedure: Trusting GPG Keys on Clients

1. On the Uyuni Server, at the command prompt, check the contents of the /srv/www/htdocs/pub/ directory. This directory contains all available public keys. Take a note of the key that applies to the client you are registering.
2. Open the relevant bootstrap script, locate the ORG_GPG_KEY= parameter and add the required key. You do not need to delete any previously stored keys. For example:

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

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

5.3.4. Create Activation Key for a Branch Server and the Retail Terminal Images

The branch server is based on the Uyuni Proxy. Its activation key must contain these child channels:

- openSUSE Leap 16.1 Updates (x86_64)
- Uyuni Client Tools for openSUSE Leap 16.1 (x86_64)
- Uyuni Proxy Stable for openSUSE Leap 16.1 (x86_64)

The activation key for retail terminal images based on openSUSE Leap 16.1 must contain these child channels:

- openSUSE Leap 16.1 Updates (x86_64)
- Uyuni Client Tools for openSUSE Leap 16.1 (x86_64)

For more information about creating activation keys, see **Client Configuration Guide › Activation Keys**.

5.3.5. Register the Branch Server and Terminals as Clients

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

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

For more information on registering your clients, see **Client Configuration Guide › Registration Overview**.

5.3.5.1. Register the Branch Server

A retail branch server is registered as an openSUSE proxy. The proxy can be bootstrapped using the Web UI, or at the command prompt. Ensure you use the activation key you created for the proxy.

- For more information about proxies, see **Installation and Upgrade Guide › Container Deployment › Proxy Container Setup Uyuni**.
- For more information about activation keys, see **Client Configuration Guide › Activation Keys**.

Procedure: Setting Up the Uyuni Proxy

1. Check that the Uyuni Proxy Stable for openSUSE Leap 16.1 (x86_64) channel is assigned to the proxy on the system profile page.

2. At the command prompt on the proxy, as root, install the proxy pattern:

```
zypper in -t pattern uyuni_proxy
```

3. Finalize the proxy setup:

```
configure-proxy.sh
```

```
`configure-proxy.sh` is an interactive script.
```

4. OPTIONAL: If you want to use the same system also as a build host, navigate to the client's system profile and check OS Image Build Host as a Add-On System Types.

5. Configure the proxy to become a branch server. On the Uyuni for Retail Server, for example, run:

```
retail_branch_init <branch_server_minion_id> --dedicated-nic eth1 \
    --branch-ip 192.168.7.5 \
    --netmask 255.255.255.0 \
    --dyn-range 192.168.7.100 192.168.7.200 \
    --server-domain branch.example.org \
    --branch-prefix uyuni
```

```
For additional options,  use the `retail_branch_init --help` command.
```

5.4. Set Up the Uyuni for Retail Environment

To set up the Uyuni for Retail environment, you will need to have already installed and configured:

- Uyuni Server
- one or more Uyuni for Retail branch server containerized proxy
- one or more Uyuni build hosts
- network stack depending on selected network architecture, see **Retail Guide › Retail Network Arch**

This section covers how to configure your Uyuni for Retail environment, including:

- Prepare POS images
- Prepare system groups for Saltboot
- Configure services for Saltboot

The very first time you set up the Uyuni for Retail environment, you will need to perform all configuration steps in order. You will need to revisit some of these steps later on as you are working with Uyuni for Retail.

For example, the first time you configure the branch server group, you will need to have images prepared already. If you are configuring more than one branch server, the same images are reused across different branch server groups.

If you have an existing environment, and need to build new images, you do not need to re-initialize the branches.

5.4.1. Prepare and Build Terminal Images

For information about Uyuni image building, see **Administration Guide › Image Management**.

Uyuni for Retail POS images are images specifically tailored for Uyuni for Retail environment and designed to be deployed using PXE booting mechanism.

5.4.1.1. POS Image Templates

As starting point, SUSE provides basic templates at https://github.com/SUSE/manager-build-profiles/tree/master/OSImage. These templates need to be adapted for specific usecases, for example by including specific applications, configuration settings, and users.



POS image templates and resulting images do not set a system user password. You will not be able to login as a system user to a system that has been installed with a SUSE provided template without customization. However you can use Salt to manage clients without a system user password. You can use Salt to set up a system user password after the terminal has been deployed.

5.4.1.2. Validate POS Image Registration

After POS image is successfully built and inspected by Uyuni, you can validate that POS boot image was correctly registered by following procedure.

Procedure Validate POS boot image

1. On Uyuni command line execute following command:

```
mgrctl exec cobbler distro list
```

2. Your POS image should be listed with an organization name appended after it.

5.4.2. Branch Identification

For every branch server choose its branch id.

As a branch id select any alphanumerical string with up to 64 characters.

5.4.3. Required System Groups

Uyuni for Retail requires:

- branch system group for every branch server proxy, using branch id as its name
- hardware type system group for every used hardware type, using HWTYPE: prefix in its name

For more information about hardware type groups, see **Retail Guide › Retail Deploy Terminals**.



Missing mandatory system group will cause terminal bootstrap to fail.

Uyuni for Retail also recognizes two optional groups for better overview:

- TERMINALS
- SERVERS

You can create system groups using the Uyuni Web UI. Navigate to **Systems › System Groups** and click **[Create System Group]**.

For more information about system groups, see **Reference Guide › Systems › System Groups**.

During terminal bootstrap, terminal automatically joins:

- branch system group based on received branch_id. This will make branch group formulas available to the terminal.
- HWType group based on SMBios information received from terminal. This will make Saltboot partitioning pillar available to the terminal.
- TERMINALS if this group exists.



In case you plan to use the branch server container host as a monitoring server with Prometheus, be aware that Prometheus demands additional hardware resources. For more information about installing Prometheus, see **Administration Guide › Monitoring**.



In case you plan to use the branch server container host with Ansible software, be aware that Ansible demands additional hardware resources. For more information about installing Ansible, see **Administration Guide › Ansible Integration**.

5.4.3.1. Saltboot group



Before configuring Saltboot group, make sure you already have at least one POS image available.

Containerized Uyuni for Retail is configured using so called Saltboot Group.

Saltboot groups are branch groups, system group with branch id as its name created in previous step, with Saltboot Group formula enabled.

Saltboot Group formula is a successor of Branch Network formula, PXE formula and TFTP formula previously used in Uyuni for Retail setups.

To connect Saltboot Group with containerized proxy fill Image Download Server entry with Fully Qualified Domain Name (FQDN) of the containerized proxy.

5.4.4. Configure Network Services for Saltboot

Saltboot technology is used to deploy POS images to the terminals. Saltboot consists of saltboot enabled initrd (build as part of POS images) and saltboot Salt states.

This section covers general information about generic Saltboot requirements. For configuration examples, see **Retail Guide › Example Configurations**.

5.4.4.1. Enable PXE Network Boot in the Terminal Network

Saltboot is usually used in network boot environment. For this to work DHCP service for the network terminal is connected to must have PXE or sometimes called BOOTP support enabled.

Listing 1. Example of ISC DHCP server configuration with PXE booting enabled

```
if substring (option vendor-class-identifier, 0, 10) = "HTTPClient" {
  option vendor-class-identifier "HTTPClient";
  filename "<FQDN of branch server proxy>/saltboot/shim.efi";
}
else {
  if option arch = 00:07 {
     filename "grub/shim.efi";
     next-server <IP address of branch server proxy>;
  }
  else {
    filename "grub/grub.0";
    next-server <IP address of branch server proxy>;
  }
}
```

Notice two important options, next-server which is set to the branch server IP address and filename set to the pxelinux.0 for BIOS based system and grub/shim.efi for UEFI systems with SecureBoot support.



Uyuni for Retail branch proxy uses different filename then previous non-containerized branch server.

For containerized branch proxy set filename to the pxelinux.0 for BIOS based system and grub/shim.efi for UEFI systems with SecureBoot.

5.4.5. Terminal Partitioning and Image Selection

Saltboot requires instructions how to partitioning terminal harddisk and what image to deploy. This is done individually for each hardware type of terminals. For more information about hardware types and partitioning, see **Retail Guide › Retail Deploy Terminals**.

Above mentioned steps are mandatory minimum for successful Saltboot deployment. For configuration examples, see **Retail Guide › Example Configurations**.

5.4.6. Synchronize Images to the Branch Server

Uyuni for Retail 2026.11 no longer need manual image synchronization, all images are available to all clients automatically.

This may not be always desired, for example to allow images gradual deployments across all branches. For a way how to limit image deployment, see **Retail Guide › Retail Best Practices**.

Chapter 6. Deploying Terminals

This section covers how to integrate terminals into your Uyuni for Retail environment. You can prepare the Uyuni for Retail installation for image deployment. Finally, you can deploy terminals using network boot and other methods.

6.1. Deployment basics

When you have the Uyuni Server and Branch Server set up, you are ready to deploy point-of-service terminals by following these steps:

1. Create hardware type groups
2. Assign and configure the Saltboot formula for each hardware type group
3. Deploy images to the terminals

Each procedure is detailed in this section.

For other methods of booting terminals, including using a USB device, or booting over a wireless network, see **Retail Guide › Retail Deploy Terminals Other**.

Terminals can be either x86-64 or arm64 architecture.

If you have many terminals, you can handle them with a script. For more information, see **Retail Guide › Retail Mass Config**.

Before terminals can be deployed, ensure you have prepared a Saltboot-based operating system image. For more information about building OS images, see **Administration Guide › Image Management**.



After you have registered new terminals, always check the Uyuni Web UI to ensure your terminals have connected successfully to the branch server. The terminals must not have directly connected to the Uyuni Server by mistake.

6.1.1. Create A Hardware Type Group

Each terminal requires a specific hardware type, which contains information about the product name and terminal manufacturer. However, at the beginning, the Uyuni database does not have this information. To tell Uyuni what image to deploy on each terminal, you can set hardware type groups. After you have created a new hardware type group, you can apply the Saltboot formula to the group and configure it for your environment.

Hardware types allow you to group devices according to manufacturer and device name. Then, all devices of a particular type can be managed as one.

Empty profiles can be assigned to a hardware type group either before or after registration. If an empty profile is not assigned to a hardware type group before registration, it will be assigned to group that best matches the product information available to it.

For this procedure, you will require the system manufacturer name and product name for your terminal.

Procedure: Creating a Hardware Type Group

1. Determine the hardware type group name. Prefix the name with HWTYPE:, followed by the system manufacturer name and product name, separated by a hyphen. For example:

```
HWTYPE:POSVendor-Terminal1
```

2. In the Uyuni Web UI, navigate to **Systems › System Groups**, and click the **[Create Group]** button.
3. In the Create System Group dialog, create a new system group, using the hardware type group name you determined in step one of this procedure.



Only use colons, hyphens, or underscores in hardware type group names. Spaces and other non-alphanumeric characters will be removed when the name is processed.

6.1.2. Assign and Configure the Saltboot Formula for Each Hardware Type Group

Each hardware type group must have the Saltboot formula applied.

Procedure: Assigning the Saltboot Formula

1. Open the details page for your new hardware type group, and navigate to the Formulas tab.
2. Select the Saltboot formula and click **[Save]**.
3. Navigate to the **Formulas › Saltboot** tab.
4. Configure the Saltboot formula. For more information about the Saltboot formula, see **Specialized Guides › Salt › Salt Formula Saltboot**.

6.1.3. Synchronize Images to the Branch Server

Manual synchronization of images is no longer needed with SUSE Multi-Linux Manager Retail Branch Server. It is however possible to warm up the cache if required.

Squid cache on the SUSE Multi-Linux Manager Retail Branch Server stores OS images with extremely long freshness and are not usually purged from the cache.

```
curl -o /dev/null -L https://$branchproxy/tftp/$url?org=$orgid
```

Where

- $branchproxy is a branch server fully-qualified domain name
- $url is the path to the image file shown in the WebUI image details page
- $orgid is the numerical identified of the organization owning this branch server instance

6.1.4. Deploy Images to the Terminals

When you have your bootstrap image ready, you can deploy the image to the terminals.

Procedure: Deploying Images to the Terminals

1. Power on your POS terminals.
2. The branch server will bootstrap the terminals according to the data you have provided.

6.1.5. Customize the Terminal Image Download Process

You can change the terminal boot process using Salt pillars. Two Salt pillars allow you to change the protocol and server used to download the image.

- The saltboot_download_protocol pillar specifies which protocol should be used to download the image to the terminal. This overrides the default protocol specified in the image pillar. Allowed values are http, https, ftp, or tftp.
- The saltboot_download_server pillar specifies which server to use to download the image. This overrides the default hostname specified in the image pillar.



In this example, the download server must be prepared by the image_sync state before you begin.

Example: Changing the Saltboot Image Download Protocol

This example changes the protocol used for all terminals.

1. Create the top.sls file with content:

```
base:
  '*':
    - saltboot_proto
```

2. Create the saltboot_proto.sls file with content:

```
saltboot_download_protocol: http
# can be http, https, ftp, tftp
```

3. Move these files to the container:

```
mgrctl cp top.sls server:/srv/pillar/
mgrctl cp saltboot_proto.sls server:/srv/pillar/
```

Example: Changing the Saltboot Image Download Location

This example changes the download location for all terminals on a specified branch server.

1. Create the top.sls file with content:

```
base:
  'minion_id_prefix:$branch_prefix':
     - match: grain
     - $branch_prefix
```

2. Create the $branch_prefix.sls file with content:

```
saltboot_download_server: $download_server_fqdn
```

3. Move these files to the container:

```
mgrctl cp top.sls server:/srv/pillar/
mgrctl cp $branch_prefix.sls server:/srv/pillar/
```

6.2. Deploy Terminals - Other Methods

If you are not able to boot terminals from the network, you can create a live USB image and deploy terminals using a removable USB storage device. You can also bootstrap terminals across a wireless network.



Hardware type groups must be created and images must be synchronized before continuing. For more information, see **Retail Guide › Retail Deploy Terminals**.



After you have registered new terminals, always check the Uyuni Web UI to ensure your terminals have connected successfully to the branch server, and not directly to the Uyuni Server by mistake.

6.2.1. Deploy Terminals with a Removable USB Device

If you do not want to boot terminals from the network, you can create a live USB image and deploy terminals using a removable USB storage device. This is useful if you cannot boot your terminals from the network, or if

you do not have a local Uyuni for Retail branch server providing network services.

You can prepare a bootable USB device with the image and tools required to deploy a POS terminal using a remote Uyuni for Retail branch server. You can create the bootable USB device on the branch server directly, or on the Uyuni for Retail Server.



POS devices booted using the USB device are assigned to the Uyuni for Retail branch server that created the USB device.

Procedure: Creating a Bootable USB Device

1. On the Uyuni for Retail branch server, at the command prompt, as root, create the POS image.

   You need to specify the size of the image, in megabytes.

   Ensure you allow at least 300 MB:

   ```
   salt-call image_sync_usb.create <usb image name> <size in MB>
   ```

2. Insert the USB device into the Uyuni for Retail branch server machine, and copy the image to the new location:

   ```
   dd bs=1M if=<usb image name> of=<path to usb device>
   ```

When you have the image on the USB drive, check that the terminals you want to deploy allow local booting. You can check this by editing the Saltboot formula in the Uyuni for Retail Web UI. For more information about the Saltboot formula, see **Specialized Guides › Salt › Salt Formula Saltboot**.

Procedure: Deploying Images to the Terminals using USB

1. Insert the USB device into the terminal.
2. Power on the POS terminal.
3. Boot from the USB device to begin bootstrapping.

6.2.2. Deploy Terminals over a Wireless Network

For terminals that cannot be connected directly to the physical network, you can deploy them over a wireless network. Wireless networks do not support PXE booting, so you must perform the initial booting and initialization of the wireless connection on the terminal using a USB device.

For more information about using USB devices to boot, see **Retail Guide › Retail Deploy Terminals Other**.



Bootstrapping across a wireless network could expose a security risk if you are using

> encrypted OS images. The boot initrd image and the partition that contains /etc/venv-salt-minion must be stored unencrypted on the terminal. This allows Uyuni for Retail to set up the wireless network. If this breaches your security requirements, you will need to use a separate production network, with network credentials managed by Salt, so that credentials are not stored on the terminal unencrypted.

Before you begin, you need to have created a bootable USB device. Ensure that the OS image you use to create the USB device has the dracut-wireless package included. For more information about using USB devices to boot, see **Retail Guide › Retail Deploy Terminals Other**.

When you have created the USB device, you need to provide the wireless credentials to the terminal. You can do this in a number of ways:

- Directly during image build.
- Add it to the initrd file on the branch server.
- During terminal booting, using the kernel command line.

Procedure: Providing Wireless Credentials During Image Build

1. Ensure that the dracut-wireless package is included in the image template.
2. Set the wireless credentials by creating or editing the etc/sysconfig/network/ifcfg-wlan0 file to the image template, with these details:

```
# ALLOW_UPDATE_FROM_INITRD
WIRELESS_ESSID=<wireless network name>
WIRELESS_WPA_PSK=<wireless network password>
```

   If you want to use different credentials for bootstrapping to what is used during normal operation, you can remove the ALLOW_UPDATE_FROM_INITRD line.

3. Build the image.
4. Prepare a USB device using this image, and boot the terminal. For more information about using USB devices to boot, see **Retail Guide › Retail Deploy Terminals Other**.

Procedure: Providing Wireless Credentials with initrd

1. Set the wireless credentials by creating or editing the etc/sysconfig/network/ifcfg-wlan0 file, with these details:

```
# ALLOW_UPDATE_FROM_INITRD
WIRELESS_ESSID=<wireless network name>
WIRELESS_WPA_PSK=<wireless network password>
```

2. . Copy the file to initrd on the branch server:

```
echo ./etc/sysconfig/network/ifcfg-wlan0 | cpio -H newc -o | gzip >>
/srv/saltboot/boot/initrd.gz
```

3. Check that the terminals you want to deploy allow local booting. You can check this by editing the Saltboot formula in the Uyuni for Retail Web UI. For more information about the Saltboot formula, see **Specialized Guides › Salt › Salt Formula Saltboot**.

Procedure: Providing Wireless Credentials During Terminal Boot

1. Mount the USB device on the terminal, and boot from it.
2. Append these commands to the kernel boot parameters:

```
WIRELESS_ESSID=<wireless_network_name>
WIRELESS_WPA_PSK=<wireless_network_password>
```

6.2.2.1. Change Wireless Credentials

After you have set the wireless credentials, you can change them as needed. The way to do this is different if you use one company-wide network, or if you have each branch server on its own separate network.

Procedure: Changing Wireless Credentials for Single Network

1. Rebuild the boot image with updated credentials.
2. Recreate the bootable USB device based on the new boot image.
3. Boot terminal from new USB device.

Procedure: Changing Wireless Credentials for Multiple Networks

1. In the /srv/salt/ directory, create a salt state called update-terminal-credentials.sls, and enter the new wireless network credentials:

```
/etc/sysconfig/network/ifcfg-wlan0
  file.managed:
   - contents: |
        WIRELESS_ESSID=<wireless_network_name>
        WIRELESS_WPA_PSK=<wireless_network_password>
# regenerate initrd
  cmd.run:
  - name: 'mkinitrd'
```

2. Apply the Salt state to the terminal:

```
salt <terminal_salt_name> state.apply update-terminal-credentials
```



If you are using a separate network for the boot phase, the managed file might need to be renamed, or extended to /etc/sysconfig/network/initrd-ifcfg-wlan0.

6.2.2.2. Use Multiple Wireless Networks

You can configure terminals to use a different set of wireless credentials during the boot process, to what they use during normal operation.

If you provide wireless credentials using initrd files, you can create two different files, one for use during boot called initrd-ifcfg-wlan0, and the other for use during normal operation, called ifcfg-wlan0.

Alternatively, you can use custom Salt states to manage wireless credentials with saltboot-hook.

First of all, you need to set the wireless details for normal operation. This will become the default settings. Then you can specify a second Salt state with the wireless details for use during the boot procedure.

Procedure: Using Different Wireless Credentials for Production Network

1. Write a custom Salt state named /srv/salt/saltboot_hook.sls containing the wireless details for normal operation. This Salt state is applied by Saltboot after the system image is deployed.

```
{% set root = salt['environ.get']('NEWROOT') %}
{{ root }}/etc/sysconfig/network/ifcfg-wlan0:
  file.managed:
   - contents: |
       WIRELESS_ESSID=<wireless_network_name>
       WIRELESS_WPA_PSK=<wireless_network_password>
    - require:
      - saltboot: saltboot_fstab
    - require_in:
      - saltboot: boot_system
```



The boot phase supports only WPA2 PSK wireless configuration. Salt-managed production configuration supports all features supported by all major operating systems.

6.3. Deploy Terminals and Auto-Accept Keys

You can configure Uyuni to automatically accept the keys of newly deployed terminals. This is achieved using Salt grains.



Automatically accepting keys is less secure than manually checking and accepting keys. Only use this method on trusted networks.

There are three different ways you can configure auto-signed grains:

- Configure Saltboot to send automatically signed grains once and then delete them. To do this, append the

Saltboot configuration to an existing initrd. For more information, see **Retail Guide › Retail Deploy Terminals Auto › Retail.deployterminals.auto.once**.

- Choose to keep the automatically signed grains on the Salt client. To do this, include the configuration file in the image source before the client image is built. After booting, the auto-signed grain is stored on the client as a regular Salt grain. For more information, see **Retail Guide › Retail Deploy Terminals Auto › Retail.deployterminals.auto.keep**.
- Configure Saltboot during PXE boot using kernel parameters. For more information, see **Retail Guide › Retail Deploy Terminals Auto › Retail.deployterminals.auto.pxe**.

When you have configured Saltboot using one of these methods, you need to configure the Uyuni Server to accept them. For more information, see **Retail Guide › Retail Deploy Terminals Auto › Retail.deployterminals.auto.server**.

6.3.1. Configure the server to auto-accept

When you have configured Saltboot using one of these methods, you need to configure the server to accept them. The server stores the autosign keys in a file within the /etc/salt/master.d/ directory, which is preserved in etc-salt volume. You can enable auto-signing by creating an auto-sign file that contains the key you created when you configured Saltboot.

Procedure: Configuring the server to auto-accept

1. At the command prompt of the Uyuni container host, as root, enter the server container:

```
mgrctl term
```

2. Inside the container, execute the following steps:

   a. In directory /etc/salt/master.d/, create file autosign_grains.conf and specify autosign_grains_dir:

```
echo "autosign_grains_dir: /etc/salt/autosign_grains" \
  > /etc/salt/master.d/autosign_grains.conf
```

   b. Create a file at /etc/salt/autosign_grains/autosign_key, that contains the auto-sign key you specified with Saltboot (replace <AUTOSIGN_KEY> with the key:

```
mkdir /etc/salt/autosign_grains
echo "<AUTOSIGN_KEY>" > /etc/salt/autosign_grains/autosign_key
```

For multiple keys, put each one on a new line.

For more information about configuring the server to automatically accept grains, see https://docs.saltproject.io/en/latest/topics/tutorials/autoaccept_grains.html.

6.3.2. Configure Saltboot to keep auto-signed grains

Use different procedure for SLE 15 and SLE 12.

Procedure: Configuring Saltboot to keep auto-signed grains (SLE 15)

1. In the location where the image source is built, such as a build host or source repository, create a configuration file called etc/venv-salt-minion/minion.d/autosign-grains.conf.
2. Edit the new configuration file with these details. You can use any value you like as the auto-sign key:

```
# create the grain
grains:
    autosign_key: <AUTOSIGN_KEY>

# send the grain as part of auth request
autosign_grains:
    - autosign_key
```

Procedure: Configuring Saltboot to keep auto-signed grains (SLE 12)

1. In the location where the image source is built, such as a build host or source repository, create a configuration file called etc/venv-salt-minion/minion.d/autosign-grains.conf. This must be outside of the root directory provided by the template. This way you prevent the inclusion of unwanted files in the initrd.
2. Edit the new configuration file with these details. You can use any value you like as the auto-sign key:

```
# create the grain
grains:
    autosign_key: <AUTOSIGN_KEY>
```

```
# send the grain as part of auth request
autosign_grains:
    - autosign_key
```

3. Create a tarball of this directory:

```
tar -czf autosign-grains.tgz etc
```

4. Edit the config.xml template file. In the <packages type="image"> element, add:

```
<archive name="autosign.tgz" bootinclude="true"/>
```

5. Save the file and rebuild the image.

6.3.3. Configure Saltboot to auto-sign during PXE boot

Procedure: Configuring Saltboot to auto-sign during PXE boot

1. Configure the PXE formula to specify these kernel parameters during booting:

```
SALT_AUTOSIGN_GRAINS=autosign_key:<AUTOSIGN_KEY>
```

2. PXE boot the Salt client. The formula creates the ./etc/venv-salt-minion/minion.d/autosign-grains-onetime.conf configuration file and passes it to initrd.

6.4. Forced Saltboot image redeployment

Systems provisioned by Saltboot are usually redeployed or repartitioned automatically when a new image is available, or Saltboot partitioning changes.

Occasionally, however, it is needed to force Saltboot to redeploy an image or repartition disk, even when automation would not do so. For these situations, Saltboot offers three ways to force redeployment or repartitioning:

- Force Saltboot redeployment using Salt grains
- Force Saltboot redeployment using custom info values

- Force Saltboot redeployment using Saltboot API call
- Force Saltboot redeployment by custom pillar



Repartitioning of a terminal removes all data stored on the terminal hard disk, including any persistent partitions.

6.4.1. Force Saltboot redeployment using Salt grains

Saltboot redeployment grains have no side effects, and do not require any further configuration. The limitation is that terminal must be accessible by salt.

Procedure: Forcing Saltboot to redeploy image

1. On the Uyuni Server, as root, apply this Salt state at the command prompt:

```
salt $terminal_minion_id state.apply saltboot.force_redeploy
```

2. Restart the terminal to pick up the changes.

Procedure: Forcing a Saltboot to repartition the hard disk

1. On the Uyuni Server, as root, apply this Salt state at the command prompt:

```
salt $terminal_minion_id state.apply saltboot.force_repartition
```

2. Restart the terminal to pick up the changes.

6.4.2. Force Saltboot redeployment using custom info values

Saltboot custom values remove the limitation on terminal being reachable by salt, however there are configuration steps.

Custom info keys and values can be also managed using API or spacecmd command. For more information, see **Reference Guide › Spacecmd › Custominfo**.

Procedure: Create custom info key for image redeployment

1. In the Uyuni Web UI, navigate to **Systems › Custom System Info**.
2. Click Create Key to create new Custom Info Key.
3. As Key Label fill in saltboot_force_redeploy.
4. As Description fill in Force redeploy Saltboot image.

5. Click Create Key.



Creating saltboot_force_redeploy custom key is a one time operation. When created, it is available for repeated use.

Procedure: Assign custom value for image redeployment

1. Navigate to the Overview page of the system you want to redeploy.
2. Select tab Custom Info.
3. Click on Create Value.
4. From the list of available keys click saltboot_force_redeploy.
5. As Value type True.
6. Click Update Key.
7. Reboot the terminal to pick up the changes.



After terminal finishes booting, Saltboot redeployment custom info setting is automatically reset to prevent repeated redeployment.

Procedure: Create custom info key for disk repartitioning

1. Navigate to menu::Systems[Custom System Info] page.
2. Click Create Key to create new Custom Info Key.
3. As Key Label fill in saltboot_force_repartition.
4. As Description fill in Force terminal disk repartition.
5. Click Create Key.



Creating saltboot_force_repartition custom key is one time operation. Once created, it is available for repeated use.

Procedure: Assign custom value for disk repartitioning

1. Navigate to the Overview page of the system you want to redeploy.
2. Select tab Custom Info.
3. Click on Create Value.
4. From the list of available keys click saltboot_force_repartition.
5. As Value type True.
6. Click Update Key.

7. Reboot the terminal to pick up the changes.

6.4.3. Force Saltboot redeployment using Saltboot API call



After terminal finishes booting, Saltboot redeployment setting is automatically reset to prevent repeated redeployment.

An API call system.setPillar can be used to set specific Saltboot options through Salt pillar data. Uyuni Saltboot integration is using tuning-saltboot pillar category to manage Saltboot tuning, including forced redeployment or disk repartition. Using this pillar category allows Uyuni to reset Saltboot flag once the terminal is booted up.

Procedure: Forcing Saltboot to redeploy image using API call using spacecmd command

1. In the console run following the command, and replace $terminal_minion_id with the actual terminal minion id:

```
spacecmd api -- -A '$terminal_minion_id,tuning-saltboot,{"saltboot":
{"force_redeployment": "True"}}' system.setPillar
```

Procedure: Forcing Saltboot to repartition disk using API call using spacecmd command

1. In the console run the following command, replace $terminal_minion_id with the actual terminal minion id:

```
spacecmd api -- -A '$terminal_minion_id,tuning-saltboot,{"saltboot":
{"force_repartition": "True"}}' system.setPillar
```

Procedure: Check Saltboot tuning options

1. In the console run the following command, and replace $terminal_minion_id with the actual terminal minion id:

```
spacecmd api -- -A '$terminal_minion_id,tuning-saltboot' system.getPillar
```



Make sure to use tuning-saltboot as pillar category in the API call.

6.4.4. Force Saltboot redeployment by custom pillar



Pillars specified outside of Uyuni database cannot be reset automatically. Without manual intervention, the terminal will download a new image on each reboot.

Procedure: Force a Saltboot to redeploy image using Saltboot pillar

1. Create new file /srv/salt/pillar/force_redeploy.sls with content:

```
saltboot:
  force_redeploy: True
```

2. Create new file or update existing file named /srv/salt/pillar/top.sls with content:

```
base:
  '$terminal_minion_id':
    - force_redeploy
```

3. Reboot the terminal to pick up the changes.

4. After the terminal finishes booting, remove modifications made in /srv/salt/pillar/top.sls file.

If your terminal encounters a problem with the file system or the partition table, you might need to remove the partition table and reformat the terminal.

Procedure: Force Saltboot to repartition disk using Saltboot pillar

1. Create new file /srv/salt/pillar/force_repartition.sls with content:

```
saltboot:
  force_repartition: True
```

2. Create new file or update existing file named /srv/salt/pillar/top.sls with content:

```
base:
  '$terminal_minion_id':
    - force_repartition
```

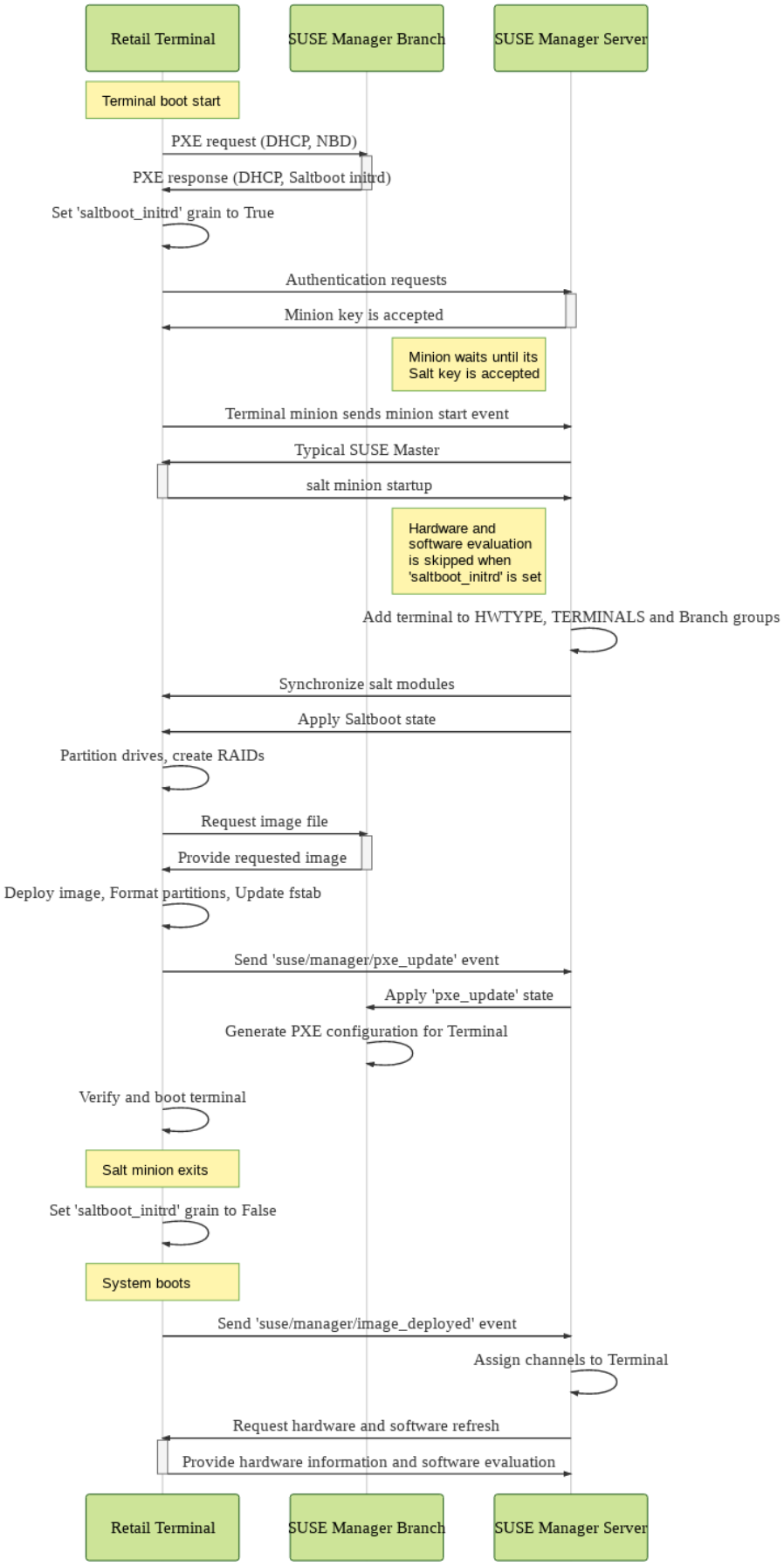
3. Reboot the terminal to pick up the changes.

4. After the terminal finishes booting, remove modifications made in /srv/salt/pillar/top.sls file.

6.5. Terminal Boot Process (Saltboot Diagram)

The Saltboot process involves the Uyuni Server, a terminal running the Saltboot initrd, and the branch server providing the Saltboot services to the terminal.

This sequence diagram explains how the three components interact with each other to boot a terminal.


Retail Terminal
SUSE Manager Branch
SUSE Manager Server
Terminal boot start
PXE request (DHCP, NBD)
PXE response (DHCP, Saltboot initrd)
Set 'saltboot_initrd' grain to True
Authentication requests
Minion key is accepted
Minion waits until its Salt key is accepted
Terminal minion sends minion start event
Typical SUSE Master
salt minion startup
Hardware and software evaluation is skipped when 'saltboot_initrd' is set
Add terminal to HWTYPE, TERMINALS and Branch groups
Synchronize salt modules
Apply Saltboot state
Partition drives, create RAIDs
Request image file
Provide requested image
Deploy image, Format partitions, Update fstab
Send 'suse/manager/pxe_update' event
Apply 'pxe_update' state
Generate PXE configuration for Terminal
Verify and boot terminal
Salt minion exits
Set 'saltboot_initrd' grain to False
System boots
Send 'suse/manager/image_deployed' event
Assign channels to Terminal
Request hardware and software refresh
Provide hardware information and software evaluation
Retail Terminal
SUSE Manager Branch
SUSE Manager Server

6.6. Terminal Names

Terminals can be named according to certain parameters, which can make it easier to match the physical device with its record in the Uyuni Web UI.

Naming schemes available are Hostname, FQDN, HWType, and MAC. Naming scheme and its configuration can be selected in the Saltboot Group formula for containerized branch servers or in the Branch Network formula for others. For more information, see **Retail Guide › Containerized Saltboot** and **Specialized Guides › Salt › Salt Formula Branchnetwork**.

By default, terminals are named according to the Hostname naming scheme with the HWType scheme as a fallback.

6.6.1. Naming by HWType

Terminal names that are derived from the hardware type use this format:

```
BranchID.Manufacturer-ProductName-SerialNumber-UniqueID
```

For example:

```
B002.TOSHIBA-6140100-41BA03X-c643
```

The BranchID is the unique identifier for the branch server that the terminal is connected to. You can disable this prefix by toggling the Do not prefix Salt client ID with Branch ID.

The Manufacturer, ProductName, and SerialNumber are provided by the terminal hardware BIOS. If the terminal does not provide a serial number, it will be omitted from the terminal name.

The UniqueID is the first four characters of a generated machine identification number. Added unique ID is a requirement for successful terminal deployment. Without unique ID, subsequent terminal registration will fail. You can disable this behavior by toggling the Do not append unique suffix to the Salt client ID.

6.6.2. Naming by Hostname

Terminal names that are derived from the hostname use this format:

```
BranchID.Hostname-UniqueID
```

For example:

```
B002.terminal-c643
```

The BranchID is the unique identifier for the branch server that the terminal is connected to. You can disable this prefix by toggling the Do not prefix Salt client ID with Branch ID checkbox.

The Hostname is the plain hostname (without domain part) of the terminal.

The UniqueID is the first four characters of a generated machine identification number. You can disable this behavior by toggling the Do not append unique suffix to the Salt client ID checkbox.

6.6.3. Naming by FQDN

Terminal names that are derived from the Fully Qualified Domain Names (FQDN) use this format:

```
BranchID.FQDN-UniqueID
```

For example:

```
B002.terminal.example.com-c643
```

The BranchID is the unique identifier for the branch server that the terminal is connected to. You can disable this prefix by toggling the Do not prefix Salt client ID with Branch ID checkbox.

The FQDN is the fully qualified domain name of the terminal.

The UniqueID is the first four characters of a generated machine identification number. You can disable this behavior by toggling the Do not append unique suffix to the Salt client ID checkbox.

6.6.4. Naming by MAC

Terminal names are derived from the network interface hardware address (MAC) and use this format:

```
BranchID.MAC-UniqueID
```

For example:

```
B0002.52:54:00:62:18:6f-cb83
```

The BranchID is the unique identifier for the branch server that the terminal is connected to. You can disable this prefix by toggling the Do not prefix Salt client ID with Branch ID checkbox.

The MAC is colon delimited hardware address of the network interface card used for booting of the terminal.

The UniqueID is the first four characters of a generated machine identification number. You can disable this behavior by toggling the Do not append unique suffix to the Salt client ID checkbox.

6.6.5. Assign Hostnames to Terminals

If you want terminal names to be derived from the hostname, you will need to ensure your terminals have a static hostname. This requires a static IP address to be able to resolve the static hostname.

There are a number of different ways to assign hostnames to terminals. This section describes how to do this when DNS and DHCP services are managed by the non-containerized branch server.

Procedure: Assigning IP Address and Hostname with Formulas

1. In the DHCP formula settings, navigate to Hosts with Static IP Address and click **[Add Item]**. For more information on the DHCP formula, see **Specialized Guides › Salt › Salt Formula Dhcpd**.
2. In the Hostname field, type the hostname of the branch server.
3. In the IP Address field, type the static IP address for the terminal. Ensure the IP address is within the range used by the branch server.
4. In the Hardware Type and Address field, type the hardware type and address in this format:

```
ethernet <terminal_MAC_address>
```

5. OPTIONAL: For multiple terminals, click **[Add Item]** and fill in the details for each terminal.
6. Click **[Save Formula]** to save the changes.
7. In the Bind formula settings, navigate to the A records of the appropriate non-reverse zone, and click **[Add Item]**. For more information on the bind formula, see **Specialized Guides › Salt › Salt Formula Bind**.
8. In the Hostname field, type the hostname of the branch server.
9. In the IP Address field, type the static IP address you assigned to the terminal in the DHCP formula settings.
10. OPTIONAL: For multiple terminals, click **[Add Item]** and fill in the details for each terminal.
11. Click **[Save Formula]** to save the changes.
12. Apply the highstate on the branch server to apply the changes.



If the terminal was previously registered using a name based on the hardware type instead of the hostname, you will need to delete the previous registration. When you re-register the terminal, use the new terminal name.

Procedure: Assigning IP Address and Hostname with YAML

1. At the command prompt on the branch server, export a YAML configuration file:

```
retail_yaml --to-yaml retail.yaml
```

2. Open the YAML file and navigate to the end of the branch server section. Add a new terminals section if it does not already exist.
3. Add the IP address, MAC address, and hardware type for the terminal, using this format:

```
$hostname:
        IP: <IP_Address>
        hwAddress: <MAC_Address>
        hwtype: <HWTYPE_Group_name_without_HWTYPE:_prefix>
```

4. Import the modified YAML file:

```
retail_yaml --from-yaml retail.yaml
```

5. Apply the highstate on the branch server to apply the changes.



If the terminal was previously registered using a name based on the hardware type instead of the hostname, you will need to delete the previous registration. When you re-register the terminal, use the new terminal name.

For more information about using YAML configuration files, see **Retail Guide › Retail Mass Config**.

6.7. Offline Use

If the Uyuni Server is offline, you can still perform some operations on the terminals. This is useful if the connection between the branch server and the Uyuni Server is unstable or has low bandwidth. This feature uses caching to perform updates.

6.7.1. Offline Terminal Reboot

If the Uyuni Server is offline, and a terminal is rebooted, it will fall back to a previously installed image.

This will occur in these situations:

- If the Saltboot formula has not started within a specified time (default value is 60 seconds).
- If the terminal does not acknowledge that the Saltboot formula has started.
- If the root partition is specified on the kernel command line (handled by the PXE formula), is mountable (and is not encrypted), and contains /etc/ImageVersion (which is created by Kiwi).

You can adjust the timeout value by changing the SALT_TIMEOUT kernel parameter. The parameter is measured in seconds, and defaults to 60.

```
SALT_TIMEOUT = 60
```

For more about kernel parameters, see **Specialized Guides › Salt › Salt Formula Pxe**.

6.7.2. Cached Terminal Updates

If the bandwidth between the branch server and the terminal is low, or for optimization of the terminal update process, POS images can be cached in advance on the terminal. The upgrade can then performed on the terminals when suitable.

This functionality requires the terminal to have a dedicated service partition. A service partition is a partition mounted as /srv/saltboot. This partition must be created before the system partition and large enough to store a POS image. To ensure that terminals will always have such a partition, use the Saltboot formula for terminal hardware type to specify the partition details. For more information, see **Specialized Guides › Salt › Salt Formula Saltboot**.

When the service partition is set up on the terminal, a POS image can be downloaded in advance by applying the saltboot.cache_image state:

```
salt $TERMINALID state.apply saltboot.cache_image
```

This can be done regularly to ensure that terminals always have an uptodate POS image downloaded.

When the terminal is rebooted and an uptodate POS image is found in the service partition, the terminal will automatically use this cached image for system redeployment.

6.8. Rate Limiting Terminals

Salt is able to run commands in parallel on a large number of terminals. This can potentially create heavy load on your infrastructure. You can use rate-limiting parameters to control the load in your environment.

For more information about rate limiting on terminals, see **Specialized Guides › Salt › Salt Rate Limiting**.

6.8.1. Troubleshooting

Sometimes when attempting to reboot a terminal after attempting to apply the Saltboot formula, the terminal will hang at the boot screen. This can be caused by a presence ping timeout value being set at a value that is too low. You can adjust the presence ping timeout value to fix this problem.

For more information about rate limiting on terminals, see **Specialized Guides › Salt › Salt Rate Limiting**.

Chapter 7. Introduction to Retail Formulas

Formulas are pre-written Salt states, that are used to configure your Uyuni for Retail installation.

You can use the Uyuni Web UI to apply common Uyuni formulas. For the most commonly used formulas, see **Specialized Guides › Salt › Salt Formulas Intro**.

All formulas must be accurately configured for your Uyuni for Retail installation to function correctly.

7.1. Branch server formulas

Branch servers are configured using formulas. Formulas can be configured using Uyuni Web UI, or the Uyuni API.

7.1.1. Containerized branch server (Uyuni 2026.11)

On a Uyuni 2026.11 Proxy, PXE, TFTP and image synchronization are provided as containerized services, which do not use formulas.

You can use other formulas on the salt-managed Proxy container host if you need them:

- Branch network formula, see **Specialized Guides › Salt › Salt Formula Branchnetwork**
- BIND formula, see **Specialized Guides › Salt › Salt Formula Bind**
- DHCPD formula, see **Specialized Guides › Salt › Salt Formula Dhcpd**

The BIND and DHCPD formulas use container images.



For air-gapped environments, the container images for BIND and DHCPD formulas must be manually pulled from a connected system, saved to a TAR archive, transferred to the air-gapped server, and then loaded. Refer to the main air-gapped deployment documentation for detailed steps on this process, ensuring you use the correct image paths and tags for the BIND and DHCPD formulas.



Badly configured formulas can result in the branch server failing to work as expected. Due to the generic nature of formulas it is difficult to do overall validation. We recommend that you configure the branch server using the Uyuni for Retail command line utilities, and use individual formula settings for further tuning if required. For more information, see **Retail Guide › Retail Install Setup**.



If a formula uses the same name as an existing Salt state, the two names will collide. This could result in the formula being used instead of the state. Always check the names of states and formulas to avoid name collisions.

When you have made changes to your formula, ensure you apply the highstate. The highstate propagates your changes to the appropriate services.

7.2. Partitioning and image deployment formula

Use the Saltboot formula to specify disk partitioning, and to select which image should be deployed. For more information about the Saltboot formula, see **Specialized Guides › Salt › Salt Formula Saltboot**.

Chapter 8. Image Pillars

If the built image type is PXE, a Salt pillar is also generated.

Image pillars are stored in the database and Salt subsystem can access details about the generated image. Details include where the image files are located and provided, image checksums, information needed for network boot, and more.

The generated pillar is available to all connected clients.

This is an example of image pillar:

```
{
  "images": {
    "POS_Image_JeOS7": {
      "7.1.0-1": {
        "url": "https://ftp/saltboot/image/POS_Image_JeOS7.x86_64-7.1.0-
1/POS_Image_JeOS7.x86_64-7.1.0",
        "arch": "x86_64",
        "hash": "7368c101e96826053c6efde0588cf365",
        "size": 1548746752,
        "sync": {
          "url": "https://manager.example.com/os-images/1/POS_Image_JeOS7-7.1.0-
1/POS_Image_JeOS7.x86_64-7.1.0",
          "hash": "7368c101e96826053c6efde0588cf365",
          "local_path": "image/POS_Image_JeOS7.x86_64-7.1.0-1"
        },
        "type": "pxe",
        "fstype": "ext3",
        "filename": "POS_Image_JeOS7.x86_64-7.1.0",
        "inactive": false,
        "boot_image": "POS_Image_JeOS7-7.1.0-1"
      }
    }
  },
  "boot_images": {
    "POS_Image_JeOS7-7.1.0-1": {
      "arch": "x86_64",
      "name": "POS_Image_JeOS7",
      "sync": {
        "initrd_url": "https://manager.example.com/os-images/1/POS_Image_JeOS7-7.1.0-
1/POS_Image_JeOS7.x86_64-7.1.0.initrd",
        "kernel_url": "https://manager.example.com/os-images/1/POS_Image_JeOS7-7.1.0-
1/POS_Image_JeOS7.x86_64-7.1.0-5.14.21-150400.24.55-default.kernel",
        "local_path": "POS_Image_JeOS7.x86_64-7.1.0-1"
      },
      "initrd": {
        "url": "https://ftp/saltboot/boot/POS_Image_JeOS7.x86_64-7.1.0-
1/POS_Image_JeOS7.x86_64-7.1.0.initrd",
        "hash": "d38b74a373bc6c9def1f069a8533d99f",
        "size": 118252253,
        "version": "7.1.0",
        "filename": "POS_Image_JeOS7.x86_64-7.1.0.initrd"
      },
      "kernel": {
        "url": "https://ftp/saltboot/boot/POS_Image_JeOS7.x86_64-7.1.0-
1/POS_Image_JeOS7.x86_64-7.1.0-5.14.21-150400.24.55-default.kernel",
        "hash": "946dac0a19125d78e282afe0e3ebf0b6",
        "size": 11444416,
```

```
        "version": "5.14.21-150400.24.55-default",
        "filename": "POS_Image_JeOS7.x86_64-7.1.0-5.14.21-150400.24.55-default.kernel"
      },
      "basename": "POS_Image_JeOS7.x86_64-7.1.0"
    }
  }
}
```

Under the Images section, there is a specific image called POS_Image_JeOS7 with a version 7.1.0-1. It provides various details such as the image URL for download, architecture, hash, size, synchronization information, image type (in this case, PXE), file system type, file name, and an inactive flag to exclude it from auto-selection. Additionally, it references the corresponding boot image named POS_Image_JeOS7-7.1.0-1.

The boot_images section contains information about the boot image POS_Image_JeOS7-7.1.0-1. It specifies the details about kernel and initrd.

Image pillar can be modified via API. This example retrieves the image pillar for a given image ID, and changes the Inactive flag to True and writes it back.

```
#!/usr/bin/env python3
from xmlrpc.client import ServerProxy

MANAGER_URL = "http://manager.example.com/rpc/api"

MANAGER_LOGIN = "admin"
MANAGER_PASSWORD = "adminpass"

IMAGE_ID=15

client = ServerProxy(MANAGER_URL)
key = client.auth.login(MANAGER_LOGIN, MANAGER_PASSWORD)

pillar = client.image.getPillar(key, IMAGE_ID)
[(name, version_dict)] = pillar['images'].items()
[(version, image_data)] = version_dict.items()
image_data['inactive'] = True
client.image.setPillar(key, IMAGE_ID, pillar)

client.auth.logout(key)
```

Chapter 9. Administration

This sections contains notes on administering your Uyuni for Retail installation. For general administration tasks, see the Uyuni documentation at https://documentation.suse.com/multi-linux-manager/.

9.1. Mass Configuration

Mass configuration is possible with branch servers and terminals.



Mass configuration tool was not yet adapted for containerized saltboot workflow.

9.1.1. Branch Server Mass Configuration

Branch servers are configured individually using formulas. If you are working in an environment with many branch servers, it often helps to configure branch servers automatically with a pre-defined configuration file, rather than configuring each one individually.



Before working with the mass configuration tool, back up the existing branch servers configuration.

The mass configuration tool overwrites the existing configuration with data specified in the provided YAML file.

The mass configuration tool does not support all possible formula configurations. Always make sure on a small sample that the mass configuration tool can configure systems as expected.

9.1.1.1. Configure Multiple Branch Servers

Configuring multiple branch servers requires the python-susemanager-retail package, which is provided with Uyuni for Retail. Install the python-susemanager-retail package on the Uyuni server.

Procedure: Configuring Multiple Branch Servers

1. Create a YAML file describing the infrastructure you intend to install. For an example YAML file, see **Retail Guide › Retail Mass Config Yaml**.
2. On a clean Uyuni for Retail installation, import the YAML file you have created:

```
retail_yaml --from-yaml filename.yaml
```

See the retail_yaml --help output for additional options.

3. In the Uyuni Web UI, check that your systems are listed correctly. Also check that the formulas you require are available.
4. Create activation keys for each of your branch servers, register them using bootstrap, and configure them as proxy servers.
5. In the States tab, click **[Apply Highstate]** to deploy your configuration changes for each branch server.

If you need to change your configuration, you can edit the YAML file at any time, and use the retail_yaml --from-yaml command to upload the new configuration.

Use empty profiles together with activation keys to onboard all the systems of your infrastructure. Use an activation key to assign the channels listed in **Retail Guide › Retail Install Setup**.

9.1.2. Terminal Mass Configuration

If you are working in an environment with many terminals, it often helps to configure terminals automatically with a pre-defined configuration file, rather than configuring each one individually.

You will need to have your infrastructure planned out ahead of time, including the IP addresses, hostnames, and domain names of branch servers and terminals. You will also require the hardware (MAC) addresses of each terminal.



The settings specified in the configuration file cannot always be successfully applied. In cases where there is a conflict, Uyuni will ignore the request in the file. This is especially important when designating hostnames. You should always check that the details have been applied as expected after using this configuration method.

9.1.2.1. Configure Multiple Terminals

Procedure: Configuring Multiple Terminals

1. Create a YAML file describing the infrastructure you intend to install, specifying the hardware address for each terminal. For an example YAML file, see **Retail Guide › Retail Mass Config Yaml**.
2. On a clean Uyuni installation, import the YAML file you have created:

```
retail_yaml --from-yaml filename.yaml
```

   See the retail_yaml --help output for additional options.

3. In the Uyuni Web UI, check that your systems are listed and displaying correctly, and the formulas you require are available.
4. Create activation keys for each of your branch servers, connect them using bootstrap, and configure them

as proxy servers.

5. In the States tab, click **[Apply Highstate]** to deploy your configuration changes for each branch server.
6. Connect your terminals according to your infrastructure plan.

If you need to change your configuration, you can edit the YAML file at any time, and use the retail_yaml --from-yaml command to upload the new configuration.

9.1.3. Export Configuration to Mass Configuration File

If you already have a configuration that you would like to export to a YAML file, use:

```
retail_yaml --to-yaml filename.yaml
```

This can be a good way to create a basic mass configuration file. However, it is important to check the file before using it, because some mandatory configuration entries may be missing.



When you are designing your configuration and creating the YAML file, ensure the branch server ID matches the fully qualified hostname, and the Salt ID. If these do not match, the bootstrap script could fail.

9.2. Example YAML File for Mass Configuration

You can use the retail_yaml command to import configuration parameters from a manually prepared YAML file. This section contains a YAML example file with comments.

Listing 2. Example: YAML Mass Terminal Configuration File

```
branches:
# there are 2 possible setups: with / without dedicated NIC
#
# with dedicated NIC
  branchserver1.branch1.cz:     # salt ID of branch server
    branch_prefix: branch1      # optional, default guessed from salt id
    server_name: branchserver1  # optional, default guessed from salt id
    server_domain: branch1.cz   # optional, default guessed from salt id
    nic: eth1                   # nic used for connecting terminals, default taken from hw
info in MLM
    dedicated_nic: true         # set to true if the terminals are on separate network
    salt_cname: branchserver1.branch1.cz      # hostname of salt master / broker for
terminals, mandatory
    configure_firewall: true    # modify firewall configuration
    branch_ip: 192.168.2.1      # default for dedicated NIC: 192.168.1.1
    netmask: 255.255.255.0      # default for dedicated NIC: 255.255.255.0
    dyn_range:                  # default for dedicated NIC: 192.168.1.10 - 192.168.1.250
        - 192.168.2.10
        - 192.168.2.250
# without dedicated NIC
# the DHCP formula is not used, DHCP is typically provided by a router
# the network parameters can be autodetected if the machine is already connected to SUSE
```

```
Manager
  branchserver2.branch2.cz:     # salt ID of branch server
    branch_prefix: branch2      # optional, default guessed from salt id
    server_name: branchserver2  # optional, default guessed from salt id
    server_domain: branch2.cz   # optional, default guessed from salt id
    salt_cname: branchserver2.branch1.cz      # FQDN of salt master / broker for
terminals, mandatory
    branch_ip: 192.168.2.1      # optional, default taken from MLM data if the machine is
registered
    netmask: 255.255.255.0      # optional, default taken from MLM data if the machine is
registered
    exclude_formulas:           # optional, do not configure listed formulas
      - dhcp                    # without dedicated NIC the dhcp service is typically
provided by a router
    hwAddress: 11:22:33:44:55:66 # optional, required to connect pre-configured entry with
particular machine
                                 # during onboarding
    terminals:                       # configuration of static terminal entries
      hostname1:                     # hostname
        hwAddress: aa:aa:aa:bb:bb:bb # required as unique id of a terminal
        IP: 192.168.2.50             # required for static dhcp and dns entry
        saltboot:                    # optional, alternative 1: configure terminal-
specific pillar data
          partitioning:              # partitioning pillar as described in saltboot
documentation
            disk1:
              device: /dev/sda
              disklabel: msdos
              partitions:
                p1:
                  flags: swap
                  format: swap
                  size_MiB: 2000.0
                p2:
                  image: POS_Image_JeOS6
                  mountpoint: /
              type: DISK
      hostname2:                     # hostname
        hwAddress: aa:aa:aa:bb:bb:cc # required as unique id of a terminal
        IP: 192.168.2.51             # required for static dhcp and dns entry
        hwtype: IBMCORPORATION-4838910 # optional, alternative 2: assign the terminal to
hwtype group
        # if neither of hwtype nor saltboot is specified, a group is assigned according to
hwtype
        # reported by bios on the first boot
hwtypes:
  IBMCORPORATION-4838910:      # HWTYPE string (manufacturer-model) as returned by bios
    description: 4838-910       # freetext description
    saltboot:
      partitioning:             # partitioning pillar as described in saltboot
documentation
        disk1:
          device: /dev/sda
          disklabel: msdos
          partitions:
            p1:
              flags: swap
              format: swap
              size_MiB: 1000.0
            p2:
              image: POS_Image_JeOS6
              mountpoint: /
          type: DISK
  TOSHIBA-6140100:
    description: HWTYPE:TOSHIBA-6140100
```

```
    saltboot:
      partitioning:
        disk1:
          device: /dev/sda
          disklabel: msdos
          partitions:
            p1:
              flags: swap
              format: swap
              size_MiB: 1000.0
            p2:
              image: POS_Image_JeOS6
              mountpoint: /
          type: DISK
```

9.3. Delta Images

If you have very large images that you need to synchronize to the branch server, you can use delta images to save network bandwidth.

A delta image contains only the differences between two regular images. If there are only a few changes between two images, the delta image can be very small. Synchronizing a delta image to the branch consumes less network bandwidth but it requires some extra hardware resources on the branch server to rebuild the installable image.

9.3.1. Building Delta Images

The retail_create_delta tool creates a delta image on the Uyuni server. The tool uses xdelta3 internally.

Use the name and the version strings of the target and the source image as parameters to the command. The format of the parameters must be <NAME>-<VERSION> and they must correspond to the image names and versions available in the pillar. For example, if the pillar contains:

```
images:
  POS_Image_JeOS6:
    6.0.0:
      ...
    6.0.1:
      ...
  POS_Image_Graphical6:
    6.0.0:
      ...
```

Then the retail_create_delta command is:

```
retail_create_delta POS_Image_JeOS6-6.0.1 POS_Image_JeOS6-6.0.0
```

This command will generate the delta image between version 6.0.0 and version 6.0.1. The resulting delta file is saved in /srv/www/os-images and the corresponding pillar in the database.

9.3.2. Tuning Delta Generation

Performance tuning is possible with the -B <VALUE> option, which is passed to xdelta3. With higher values you achieve smaller deltas at the cost of higher memory requirements. For more information, see the xdelta3 documentation (man xdelta3).

9.3.3. Image Synchronizing to the Branch Server

When an image is synchronized to the branch server, the image-sync-formula first checks whether the source image is available on the branch server. Only if the source image is available, the delta will be downloaded to save network bandwidth.

Chapter 10. Upgrade Uyuni for Retail Branch Server

This section describes upgrading the Uyuni for Retail Branch Server to the next SP (service pack).

The Uyuni for Retail Branch Server is a client system similar to the Uyuni Proxy, with additional Uyuni for Retail features.



Upgrade the Uyuni Server before starting the Uyuni for Retail upgrade.

Procedure: Upgrading the Uyuni for Retail Branch Server

1. For general information about upgrading a proxy client, see the release notes and the proxy sections of the Installation and Upgrade Guide.
2. After the proxy upgrade is complete, apply the highstate on the Uyuni for Retail Branch Server. When applying the highstate, the retail functionality will also be updated.

Chapter 11. Example configurations

This section contains some example configurations using various configuration techniques.

11.1. Configurations Using Uyuni for Retail Branch Proxy 5.0 and Later

- **Retail Guide › Containerized Saltboot**

  Configuration of Saltboot using containerized proxy.

11.2. Configurations Using Uyuni for Retail Branch Proxy 4.3

- **Retail Guide › Dedicated with Formulas 43**

  Manual configuration of all branch services using formulas with forms to configure Saltboot with dedicated network for POS terminals.

- **Retail Guide › Dedicated with Scripts**

  Automatic configuration of all branch services using Uyuni for Retail scripts to configure Saltboot with dedicated network for POS terminals.

- **Retail Guide › Shared Central Dns**

  Configuration of basic branch services using formulas with forms to configure Saltboot in shared network environment.

- **Retail Guide › Containerized Saltboot**

  Configuration of Saltboot using containerized proxy.

11.3. Set Up the Uyuni for Retail Environment using containerized proxy

To set up the Uyuni for Retail environment, you will need to have already installed and configured:

- Uyuni for Retail Server
- one or more Uyuni for Retail containerized proxies
- one or more Uyuni build host

This section covers how to configure your environment using containerized Uyuni Proxy for Saltboot

deployment.



Containerized workflow not longer implicitly configures DHCP for PXE booting. Without the formula, make sure your DHCP server has correct PXE booting setting pointing to containerized proxy. For more information how to use DHCP formula to configure DHCP server, see **Specialized Guides › Salt › Salt Formula Dhcpd**.

11.3.1. Requirements and assumptions

- Containerized workflow requires POS images build using the newest Uyuni Server.
- In this example we are going to use branch id `B0001`.
- As a terminal we assume to have one terminal with hardware manufacturer `TerminalOEM` and model `T1000`.
- For POS image we assume to have one with name `POS_Image_JeOS7`.

11.3.2. Create required system groups

Procedure

1. Create the following three system groups. For guidelines on how to create the groups, see **Reference Guide › Systems › System Groups**.
    - TERMINALS
    - HWType:TerminalOEM-T1000
    - `B0001`

    The first group is generic optional group for collecting all POS terminals. The second group is hardware type group for our POS terminal. The third group is mandatory branch group.

    For more information about Saltboot groups, see **Retail Guide › Retail Install Setup**.
2. Assign `Saltboot Group` formula to group `B0001` we just created. This converts the branch group to `Saltboot Group`.

11.3.3. Saltboot group

Containerized Uyuni for Retail is configured in `Saltboot Group`.

Saltboot groups are branch groups, system group with `branch id` as its name and `Saltboot Group` formula enabled.

Saltboot Group formula is a successor of Branch Network formula, PXE formula and TFTP formula used in regular Uyuni for Retail setups.

Name of the Saltboot Group is automatically used as a branch id, an identifier for group of machines booted through particular containerized Uyuni Proxy.

All Saltboot deployed machines though containerized proxy will automatically became members of its Saltboot group.

To connect Saltboot group with containerized proxy, fill Image Download Server entry with Fully Qualified Domain Name (FQDN) of the containerized proxy.

With this, mandatory configuration is finished. The rest of configuration is optional.

11.3.3.1. Default boot image

Configure Default boot image for new registrations to specify what boot image should be booted by not yet registered POS terminal.

This is useful when stable boot image is wanted for initial deployments. Without this setting, newest built boot image is used as default boot image.

If Default boot image for new registrations is set, option to set its version appears under name Default boot image version where specific image version can be set.

11.3.3.2. Kernel option for the Saltboot group

Option Kernel parameters for the group can be used to pass extra kernel options to all POS terminals registered withing this Saltboot group.

11.3.3.3. Naming scheme for new registrations

Last three options are related to how will be newly registered machine visible in Uyuni Server.

For explanation of possible configurations, see **Retail Guide › Retail Terminal Names**.

11.3.4. Comparing containerized and non-containerized workflows

External DHCP service must be used with containerized Saltboot.

For more information about how to enable PXE booting in DHCP service, see **Retail Guide › Retail Install Setup**.

Containerized workflow relies on updated image building in Uyuni Server, where PXE images are no longer collected as bundle, but kernel, initrd and filesystem image are collected individually.

Containerized workflow uses new TFTP container which instead of providing files present on the proxy, routes TFTP requests as HTTP requests through local proxy to Uyuni Server.

Containerized proxy is not a Salt client, it is not possible to call image-sync state.



Once POS image is built and made available on Uyuni Server, it is immediately available to the Saltboot clients as well. Image synchronization is not needed, nor available. This may have implications on how images are deployed to production.

The following sections differentiate between containerized and regular workflow. Both are assuming proxy (containerized or in form of Uyuni for Retail Branch Server) are available.

Procedure: Deploying with containerized workflow

1. Build POS image
2. Configure DHCP server for PXE booting for given network
3. Create Saltboot group and configure it for existing containerized proxy
4. Boot system to be deployed

Procedure: Deploying with non-ontainerized workflow

1. Build POS image
2. Configure and apply Retail formulas on Uyuni for Retail Branch server
3. Apply highstate state on the Branch server
4. Create branch group with the name of Branch ID as set in retail formulas
5. Apply image-sync state on configured Uyuni for Retail Branch server
6. Boot system to be deployed

11.3.5. Validating Saltboot group configuration

Containerized Saltboot utilizes Cobbler system underneath for managing PXE and UEFI configuration.

When new PXE image is built (such as Uyuni for Retail POS_Image_JeOS images) cobbler distro and cobbler profile are automatically generated for this image.

For example, when first image POS_Image_JeOS version 7.0.0 is build under organization with number 1 cobbler list will show:

```
# cobbler list

distros:
   1-POS_Image_JeOS7-7.0.0-1

profiles:
   1-POS_Image_JeOS7-7.0.0-1
```

These entries contain information about kernel and initrd. These entries are however not yet available for PXE booting.

Only when Saltboot group is created, new Cobbler profile is created for this Saltboot group which points to cobbler distro based on default boot image configuration.

For example, when system group B0001 is created and Saltboot group formula is assigned and configured for this group, new Cobbler profile is created.

```
# cobbler list

distros:
   1-POS_Image_JeOS7-7.0.0-1

profiles:
   1-POS_Image_JeOS7-7.0.0-1
   1-B0001
```

When inspecting this new group using command cobbler profile report --name 1-B0001 details of this profile reveal configuration of this Saltboot group.

```
# cobbler profile report --name 1-B0001

Name                           : 1-B0001
Comment                        : Saltboot group B0001 of organization SUSE default profile
Distribution                   : 1-POS_Image_JeOS7-7.0.0-1
Kernel Options                 : {'MASTER': ['downloadserver.example.org'],
'MINION_ID_PREFIX': ['B0001']}
```

Kernel options in example are always present and are internal for Saltboot functionality.

With this information Cobbler is able to generate required PXE and UEFI Grub configurations which can be checked in /srv/tftpboot/pxelinux.cfg/default and /srv/tftpboot/grub/x86_64_menu_items.cfg.

These files contain the end result which will be used by PXE client when determining what to boot and with which parameters.



If all built OS images are removed while a branch Saltboot group is configured, the corresponding Cobbler profile for this group will also be removed. This is expected behavior. However, when a new image is subsequently built, a new Cobbler entry is not automatically generated for the group, resulting in a missing entry in the boot menu.

> To recreate the required Cobbler entry, you must open and resave the Saltboot Group formula.

Chapter 12. Best practices

12.1. Deploying new images

This file describes how to deploy a new image on selected terminals first, test it, and then deploy it to all terminals.

The hardware type group (i.e. HWTYPE group) is used for booting new terminals, while already deployed terminals can be moved to any group that provides the Saltboot formula. This flexibility allows for testing of new images or configurations.

Procedure: Dual group image deployment

1. Build new image. Use image synchronization formula to synchronize the image to all Branch Servers.
2. Create a test group.
    - Name the group, for example, TEST:POSVendor-Terminal. The name can be chosen freely.
    - Add the Saltboot formula to this group. Use the same configuration as in the original hardware type group (HWTYPE), except for the image version. Use the name and version of the new image.
3. Move selected terminals.
    - Identify the terminals you want to test the new image on.
    - Transfer these terminals from the group HWTYPE:POSVendor-Terminal group to the newly created group TEST:POSVendor-Terminal. It is recommended to use System Set Manager for this task.
4. Reboot terminals.
    - Power on the selected POS terminals. They should boot the new image.
5. Verify and evaluate.
    - Monitor the terminals in the TEST:POSVendor-Terminal group.
    - Check if the new image performs as expected and if there are any issues or errors encountered.
6. Update HWTYPE group.
    - Once you have confirmed that the new image works correctly, the HWTYPE group can be updated.
    - Modify the image version in the HWTYPE:POSVendor-Terminal group to match the tested and approved version.
    - This ensures that all other terminals receive the new image.

12.1.1. Controlling image versions

By default, Saltboot formula selects the specified image name and version or selects the highest one if the

version is not specified.

There are multiple methods to control the image version within the HWTYPE and TEST groups:

- Specify exact version.
  - In both the HWTYPE and TEST groups, specify explicitly the exact version of the image to be deployed.
  - Make sure that the specified image version is already synchronized.
- Mark new image as Inactive.
  - In the HWTYPE group, do not specify the image version.
  - Mark the new image as Inactive after building and before synchronizing it to Branch Server.
  - The Inactive flag ensures, that the image is not auto-selected when the image version is not specified.
  - After testing, remove the Inactive flag so the image can be auto-selected in the HWTYPE group.
  - The Inactive flag can be accessed through the API. For more information, see **Retail Guide › Retail Image Pillars**.
- Utilize Freeze Image flag.
  - In the HWTYPE group, omit the image version.
  - Before synchronizing the new image, set the Freeze Image flag in Saltboot formula in HWTYPE group. This ensures that the already deployed terminals in this group will keep the old image.
  - Synchronize and test the new image in the TEST group.
  - Reset the Freeze Image flag in HWTYPE group, so the terminals in this group will be updated to the new image too.



Do not use the whitelist in Image Synchronization formula to control which terminal boots which image. It does not provide the required granularity and does not work with containerized Branch Server.

Chapter 13. What Next?

This document covers only a sub-section of information about your Uyuni for Retail installation. If you need further information or support, try one of these options.

13.1. More Documentation

For Uyuni documentation, visit https://documentation.suse.com//4.3/.

Legacy documentation is available upon request for information purposes only. Note, however, that Uyuni for Retail documentation supersedes legacy information.

13.2. Support

For personalized support, log in to your SUSE Customer Center account at https://scc.suse.com/login.

For assistance with planning and installing your Uyuni for Retail environment, contact the SUSE Consulting team.

Chapter 14. 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 "GNU Free Documentation License".