SUSE

SUSE Manager 4.3

Specialized Guides

Contents

Specialized Guides Overview

Updated: 2025-07-21

This book contains a list of special topics and functionalities of SUSE Manager.

It is designed to introduce basic, routine or some advanced tasks, by explaining what you are achieving in each step, and the various options available to you along the way.

You can read specialized guides for:

- **Specialized-guides › Public-cloud-guide**
- **Specialized-guides › Salt**
- **Specialized-guides › Large-deployments**
- **Specialized-guides › Qs-sap**

Chapter 1. Public Cloud Guide

Updated: 2025-07-21

This guide shows you the fastest way to get SUSE Manager up and running in a public cloud environment using on-demand, Pay-as-you-go (PAYG), or BYOS services. Additionally, it assumes that you are installing the SUSE Manager Server on a single cloud instance. These procedures have been tested on Amazon Web Services, Microsoft Azure, and Google Cloud Compute.

Pay-as-you-go or PAYG services currently work on both Amazon Web Services and Microsoft Azure.

For more information on using SUSE Manager, see the official SUSE Manager documentation at https://documentation.suse.com/suma.

1.1. BYOS Guide

1.1.1. Bring your own subscription (BYOS)

Bring your own subscription (BYOS) images are useful if you already have a support contract with SUSE and want to move your workloads to the public cloud. An instance launched from a BYOS image is equivalent to a physical machine that just received a SUSE Linux Enterprise Server installation from the SUSE Linux Enterprise Server installation image.

After creation:

- use **registercloudguest** to register the instance with the SUSE-operated update infrastructure in the cloud framework, or
- use the command **SUSEConnect** to register the system with SUSE Customer Center using the entitlements you already have.

Obtaining a SUSE Manager Server Subscription for BYOS

To obtain a SUSE Manager Server Lifecycle subscription, customers must send an email to susemanagerkey@suse.com, including proof of running at least 10 instances of SLES for SAP BYOS. Proof can be a screenshot of the relevant section of the cloud console. The subscription key is valid for one year, and customers can renew it by providing updated proof of their instances at the end of the period.

Connect your system to your own running RMT or SUSE Manager infrastructure in the same way you connect systems in your data center.

BYOS instances make it easier to manage extensions such as **LTSS** or **kernel live-patching**. Extensions for PAYG instances can only be used in conjunction with SUSE Manager.

1.1.2. Initial setup

This guide shows you the fastest way to get SUSE Manager up and running in a public cloud using on-demand or BYOS services. We have tested using SUSE Manager on Amazon EC2, Google Compute

Engine, and Microsoft Azure, but these procedures should work with other public cloud providers as well, with some variation.

There are three main methods of using SUSE Manager on a public cloud service.

Bring your own subscription (BYOS)

Most public cloud providers make a BYOS image of SUSE Manager available. This means you do not need to install SUSE Manager, just set up the server. You will need to have SUSE product entitlements before you begin, and there are some additional setup steps required. The public cloud documentation in the SUSE Manager documentation suite assumes you are using this method.

Virtual machine on public cloud

In this method, you subscribe to a public cloud service, and install SUSE Manager in a virtual machine, using the Unified Installer. You will need to have SUSE product entitlements before you begin. You can do this by following all the same instructions as you would for any local SUSE Manager installation.

On-demand SUSE Linux Enterprise Server pay as you go (PAYG) image

Most public cloud providers make SUSE Linux Enterprise Server available as a BYOS image. This means that SUSE Linux Enterprise Server is pre-installed, and you can install SUSE Manager on top, using the Unified Installer. You can do this by following all the same instructions as you would for any local SUSE Manager installation. You will need to have SUSE product entitlements before you begin. Be careful with this method, because you might end up requiring additional product entitlements that could drive up your costs.

If you are using the BYOS method, start by logging in to your chosen public cloud provider, and launching a SUSE Manager instance. Depending on the public cloud you are using, you can usually locate the SUSE Manager Server BYOS images by searching for **suse manager**. In EC2, you need to search within the Community AMIs. In GCE and Azure, search the marketplace.

Select a public cloud instance type that meets the hardware and networking requirements in **Installation-and-upgrade › Pubcloud-requirements**.

When you have your virtual machine ready, and the BYOS image installed, you need to set up SUSE Manager. This is done in the same way as a local installation, using YaST. When you have completed SUSE Manager setup, you need to activate the public cloud module. You can then complete setup in the SUSE Manager Web UI. For more information about setting up, see **Installation-and-upgrade › Pubcloud-setup**.

1.1.3. Register clients

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

For instructions on registering clients on a public cloud, see **Client-configuration › Clients-pubcloud**.

For instructions on connecting PAYG instances, see **Installation-and-upgrade › Connect-payg**.

Additional Resources

For more information on using SUSE products in the public cloud see: SUSE Linux Enterprise Public Cloud Guide.

For more SUSE Manager product documentation, see https://documentation.suse.com/suma.

To raise an issue or propose a change to the documentation, use the links under the **Resources** menu on the documentation site.

1.2. PAYG Guide

1.2.1. Overview of PAYG

Introduction to PAYG

SUSE Manager PAYG or Pay-as-you-go is a flexible and cost-effective solution that allows users to manage and monitor their systems from several cloud services. Unlike traditional subscription models, PAYG does not require a long-term subscription through the SUSE Customer Center. Users can leverage images of SUSE Manager on a Pay-as-you-go basis, paying only for the time of use, and the number of **managed** and **monitored** systems. PAYG with SUSE Manager is an optimal solution for those seeking control, convenience, and cost savings when managing and monitoring their infrastructure on AWS, Azure, and GCP.

1. Obtaining a SUSE Manager Server Subscription (optional)

For customers running 10 or more PAYG SLES for SAP Applications instances who also wish to access the SUSE Manager (SUMA) BYOS Lifecycle subscription, an email must be sent to susemanagerkey@suse.com. The email should include proof of at least 10 active SLES for SAP PAYG instances, such as a screenshot of the relevant section of their cloud console. The subscription key is valid for one year and can be renewed by providing updated proof of the instances at the end of the subscription period.

Benefits for using PAYG

The SUSE Manager PAYG instance offers a flexible solution tailored to the modern business landscape. Key benefits include:

- **Cost Efficiency:** Pay only for what you use, aligning costs with actual system management needs.
- **No Subscription Required:** Eliminate the need for ongoing commitments or long-term subscriptions, allowing for budget flexibility.
- **Ease of Use:** Utilize SUSE Manager’ s powerful features on a **PAYG** or **Pay-as-you-go** basis, streamlining initial setup and ongoing systems management.
- **Scalability:** Easily scale the number of systems managed according to your requirements, without worrying about subscription limitations.

- **Rapid Deployment:** Quickly run the image of your choice and begin managing your systems, enhancing responsiveness to your needs.
- **Integration with Cloud Services:** Leverage SUSE Manager PAYG on AWS, Azure, or GCP taking advantage of cloud infrastructure for an agile systems management experience.

1.2.2. PAYG limitations

Onboarded BYOS instances without SCC credentials

Valid SCC credentials are needed to manage Bring-your-own-subscription (BYOS) instances of SUSE products in SUSE Manager PAYG. This applies to all SUSE products except SUSE Manager Proxy BYOS. Without any SCC credentials, SUSE Manager PAYG will not allow onboarding any of the mentioned BYOS instances. The user will get an error.

There is a possibility that the user has valid SCC credentials set with already synced-up channels and the BYOS onboarded, but at some point decides to delete those credentials.

In the scenario of having already onboarded BYOS instances but no SCC credentials set, the following restrictions apply:

- A warning message will be shown to the user to let them know that they will not be able to manage those instances.
- Non-compliant BYOS instances will not be able to be managed neither through Salt or Salt-SSH. This applies to individual server management, SSM and action-chains. When dealing with several servers at once, as long as there is one non-compliant BYOS instance among them, the entire action/action-chain will be aborted for all servers.
- Non-compliant BYOS instances will not be able to access any repository hosted in SUSE Manager PAYG as long as SCC credentials are not set. SUSE Manager PAYG will return **HTTP 401 Unauthorized** until the situation is amended, that is trying to install or upgrade packages using zypper from those machines directly will not work.

1.2.3. PAYG on AWS

AWS System Requirements

When setting up a SUSE Manager PAYG instance on AWS, it’ s essential to consider system requirements for optimal performance and functionality. The default requirements outlined below have been tailored for smooth deployment and operation.

表格 1. AWS System Requirements

Requirement	Details
root storage	100 GB
spacewalk storage	500 GB
database storage	80 GB

Requirement	Details
CPU	4 cores; Use the **m5a**, **m5i**, **m6a**, **m6i** systems or newer
Memory	32 GB (minimum 16 GB)
Network Configuration	Typically configured by your organization

Obtaining the SUSE Manager Server PAYG public cloud image on AWS

Follow these step-by-step instructions to locate the SUSE Manager 4.3 PAYG image on AWS. You can also review the latest available images for the public cloud using Pint (Public Cloud Information Tracker).

See: pint.suse.com

Procedure: Obtaining the SUSE Manager PAYG image on AWS

1. In your browser navigate to the AWS Management Console.
2. Enter your AWS credentials and login.
3. From the AWS Management Console dashboard, locate the **Services** dropdown.
4. From the menu select **EC2** under the **Compute** section.
5. In the EC2 dashboard, in the main content section, you’ ll find the **[Launch Instance]** button. Click it.

From the **Choose an Amazon Machine Image (AMI)** page. Select **Browse more AMIs**

1. Search for **SUSE Manager**
2. Depending on your location select your AMI image.
   - **SUSE Manager Server Family (EU and UK only)**
   - **SUSE Manager Server Family (non-EU and non-UK-only)**.
3. Click **[Select]**.
4. Proceed for image configuration.

Initial Preparation and Configuration of the AWS Image

This section covers initial preparation and configuration of the image on AWS.

The IAM Role

To ensure seamless operation, proper AWS permissions for the IAM (Identity and Access Management) role are essential. This role must be attached to the instance, otherwise disconnection errors will be thrown after one hour.

You can pre-create the role on your AWS account and then select it from the existing **[IAM role]** drop-down during the instance configuration process.

If you create the role during the creation of an instance, the role will not be attached automatically. The role should be added post setup.

A role can be assigned to a running instance through the following steps: **Actions › Security › Modify IAM-role**. It needs to have the following policy attached in order to be used:

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AWSMarketplaceMeteringFullAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AWSMarketplaceMeteringFullAccess"
        }
    ]
}
```

To allow the **EC2** service to assume such role:

```
"AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ec2.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
```

IAM role can be attached to an **EC2** instance by means of an instance profile, which acts as a container for the role itself.

For detailed information on IAM roles, see: docs.aws.amazon.com/IAM

Configure SUSE Manager Instance

Procedure: Configuring your SUSE Manager instance

1. Select the AMI image and provide a meaningful **name** for your server. Review the details and pricing information provided.
2. You have the option to configure additional instance details in subsequent steps, such as storage, tags, security groups, and the required **IAM role**.
    a. The image automatically suggests a **default CPU configuration**.
    b. Choose an existing key pair or create a new one to grant access to the instance.
    c. Your organization should provide the necessary security groups and network configuration.
    d. If an IAM role has been previously created, you can attach it by selecting it from the existing **IAM role** drop-down. Otherwise, attach this role after the instance has been launched.
    e. Create the following partitions

- **100 GB** for the root partition
- **500 GB** for spacewalk storage
- **80 GB** for the database.

  For more information see: AWS requirements

f. Follow the prompts from AWS to complete the configuration as needed.

3. After reviewing your configuration click the **[Launch instance]** dropdown.
4. Click on the **[Launch Instances]** button.
5. You will be redirected to the **EC2** Dashboard.
6. Verify that the instance has successfully launched by selecting **Instances** on the left sidebar. The SUSE Manager PAYG instance should be running in the main content area.

Usage and Costs

Because this is a PAYG image, you will be billed according to your actual usage, including the number of systems you **manage** and **monitor** with this instance. It is essential to regularly track and review your usage to prevent unexpected costs and ensure alignment with your needs.

Initial setup

Procedure: Login and update the system

1. SSH into your AWS instance.
2. Switch to the root user and update the system.

```
sudo -i
zypper refresh
zypper update
```

3. Configure storage with the **suma-storage** tool. For help see: **suma-storage --help** This tool simplifies creating the storage disk device and database device.

Use **lsblk** to see the names of your disk devices. You will see a list of partitions that were created when we configured the AWS image.

- Root partition 100 GB
- Spacewalk storage 500 GB
- Database storage 80 GB

1. Format the command in the following manner:

```
suma-storage <storage-disk-device> [<database-disk-device>]

#For example:
```

```
suma-storage /dev/nvme1n1 /dev/nvme2n1
```

This command will create the following storage locations:

~~~
``` /manager_storage /manager_storage/spacewalk ```

``` /var/spacewalk /var/cache/rhn /var/lib/spacewalk ```
~~~

Run setup

Execute **yast2 susemanager_setup** as root.

For more information on a typical SUSE Manager installation, see **Installation-and-upgrade › Server-setup**.

SUSE Manager PAYG WebUI setup

Requirements

SUSE Manager must be setup and running on AWS. For more information on finding and setting up SUSE Manager, see **Specialized-guides › Public-cloud-guide**.

The instance-flavor-check tool and PAYG schedule check

instance-flavor-check Tool

The **instance-flavor-check** tool is an essential component that comes pre-installed by default. Its presence is crucial, as it determines the status of PAYG instances. When invoked, this tool identifies and categorizes the system as a PAYG instance. By default this tool is configured to run on a schedule named **update-payg-default**.

update-payg-default schedule task

Navigate to the **Admin › Task Schedules** to find the **update-payg-default** task. By default, this task is set to run at 10-minute intervals. Its primary function is to monitor the presence of all running PAYG instances by executing the **instance-flavor-check tool**.

Login to the Webui

Procedure: Login

Login to the SUSE Manager Server WebUI. The username is **admin** the password will be the **Instance ID** on AWS EC2.

1. Select **Admin › Setup Wizard**.

Configure the PAYG credentials and product list

On a new server, the organization credentials table is listed at **Admin › Setup Wizard › Organization Credentials**. The Products table listed under **Admin › Setup Wizard ›**

Products will be empty.

Procedure: Verify credentials for PAYG and refreshing the product list

1. On your new server navigate to the **Admin › Setup Wizard › Pay-as-you-go** tab to view and manage PAYG credentials. Localhost should provide the description **SUSE Manager Pay-as-you-go** and the status should read **Credentials successfully updated**.
2. Navigate to the **Admin › Products** tab and click **[Refresh]**.
3. Once the refresh completes the following products are listed:
   - **SUSE Manager Proxy**
   - **SUSE Manager Server**
   - **Free products**

Ubuntu and Debian

Ubuntu and Debian are not currently supported by our RMT servers.

Red Hat Linux Enterprise Clients

SUSE Manager offers support for Red Hat Enterprise Linux clients; however it is important to note that content still needs to be added manually and is not available through the cloud-based RMT service.

For more information see: **Client-configuration › Registration-overview-redhat**.

PAYG Client connections and Client Registration

Managing connections and registering PAYG and BYOS images

PAYG (Pay-As-You-Go) Images

Connecting and registering PAYG images is supported and can be done without additional prerequisites.

BYOS (Bring Your Own Subscription) Images

Registering BYOS images requires SCC (SUSE Customer Center) credentials beforehand. If a user attempts to onboard a BYOS image without providing SCC credentials, the operation will fail, as these credentials are required to verify your subscription.

- Always verify the type of image (PAYG or BYOS) prior to initiating the registration process with SUSE Manager to ensure a smooth and error-free connection.
- If you have SCC credentials ensure your SCC credentials are valid and have the necessary permissions for utilizing BYOS images.

Do not register PAYG instances with the SUSE Customer Center. Only BYOS instances should be registered against the SUSE Customer Center.

Registering PAYG instances with the SUSE Customer Center or your own RMT

server will create conflicts that are not easily solved. PAYG instances are automatically registered against the correct update server when added to your cloud service provider SUSE Manager instance.

Starting up additional PAYG SUSE products in AWS

For more information on using SUSE PAYG products such as SUSE Linux Enterprise in the public cloud see: SUSE Linux Enterprise Public Cloud Guide

Establishing a PAYG SUSE Linux Enterprise connection with SUSE Manager PAYG

It is important to recognize that there is a difference between **Pay-as-you-go connections** and **bootstrapping**. Bootstrapping is also known as **client registration**.

Once you have started up your SUSE Manager PAYG instance and refreshed products you can add **Pay-as-you-go connections** via the **Admin › Setup Wizard › Pay-as-you-go** [**+Add Pay-as-you-go**] button. This action links your SUSE Manager PAYG instance with additional PAYG products on AWS.

Bootstrapping or **client registration** is the process of onboarding clients in SUSE Manager. The registration process installs the appropriate client tools and provides full management functionality.

To manage your clients, start by **establishing connections** and then proceed to **bootstrap** your clients. These two actions are the steps required for initiating and completing the client onboarding process.

Procedure: Connecting SUSE Linux Enterprise PAYG instances to SUSE Manager PAYG

1. Return to the SUSE Manager Server WebUI.
2. Navigate to **Admin › Setup Wizard › Pay-as-you-go** and click the [**+Add Pay-as-you-go**] button.
3. Add a description.
4. Complete the instance SSH connection data:
   - Host: **CSP URL of the machine**
   - SSH Port: **22**
   - User: **root**
   - Password: **<password>**
   - SSH Private Key
5. Click [**Create**] and wait for the sync to finish.
6. Proceed to **Admin › Setup Wizard › Products** to access your PAYG SUSE Linux Enterprise products.

Registering Clients

Now that you have established connections to the SUSE Manager PAYG instance you can register them as clients. To learn how to register various clients see: **Client-configuration › Registration-methods**

1.2.4. PAYG on Azure

Azure System Requirements

When setting up a SUSE Manager PAYG instance on Microsoft Azure, it’s essential to consider system requirements for optimal performance and functionality. The default requirements outlined below have been tailored for smooth deployment and operation.

By default, certain disks are automatically generated when establishing a SUSE Manager PAYG instance on Microsoft Azure. To complete the setup of these disks, use the **suma-storage** tool.

表格 2. Azure System Requirements

Requirement	Details
root storage	100 GB
spacewalk storage	500 GB
database storage	80 GB
CPU	4 cores
Memory	32 GB (minimum 16 GB)
Network Configuration	Typically configured by your organization

Obtaining the SUSE Manager Server PAYG public cloud image on Microsoft Azure

Follow these step-by-step instructions to locate the SUSE Manager 4.3 PAYG image on Microsoft Azure. You can also review the latest available images for the public cloud using Pint (Public Cloud Information Tracker).

See: pint.suse.com

There are currently two offers for SUSE Manager PAYG on Microsoft Azure:

- **manager-server-4-3-ltd**
- **manager-server-4-3-llc**

Procedure: Obtaining the SUSE Manager PAYG image on Microsoft Azure

1. Start by logging into your Azure account at Azure Portal.
2. Once logged in, navigate to the **Virtual Machines** section. This can be found in the dashboard or by using the search bar at the top of the Azure Portal.

3. Click on **Create a Virtual Machine.** Here, you will be asked to fill in details such as the subscription, resource group, and virtual machine name.
4. During the creation of a virtual machine there is an option for **Image.** Select the correct cloud image for your VM.
5. If you don’ t find the image you need, you can browse the Azure Marketplace. Click on **See all images** or go to the **Marketplace** and search for the image you need.
6. After selecting your image, continue with the configuration of your virtual machine. Review all settings, and then create your VM.

Initial Preparation and Configuration of the Microsoft Azure Image

This section covers initial preparation and configuration of the image on Microsoft Azure.

Procedure: Configuring your SUSE Manager instance

1. Start by logging into your Azure account at Azure Portal.
2. In the Azure Portal, find the menu:[Virtual Machines] section. This can be done either through the **dashboard** or by using the **search bar**.
3. Select menu:[Create a resource] in the top left corner of the portal.
4. Choose menu:[Compute] and then select menu:[Virtual Machine].
5. You will be guided through the **Create a virtual machine** process.
6. Fill in the required details such as subscription, resource group, and VM name.
7. Choose a region for your VM. Ensure it complies with any geo-fencing policies you might have.
8. In the **Image** dropdown, select the **manager-server-4-3-ltd** or the **manager-server-4-3-llc** image for your VM. If the image is not listed, select **Browse all public and private images** to find the required image.
9. Suggested sizes are selected by default for this VM. Configure optional features such as virtual CPUs, memory as required.
10. Set up an administrator account for the VM. For SUSE Linux Enterprise Server, this can be a username and SSH public key.

Ensure the VM is connected to the appropriate virtual network (VNet) and subnet, especially if it needs to communicate with other specific services like SUSE Manager. For network requirements, see **Specialized-guides › Public-cloud-guide**.

1. Configure additional settings such as storage, monitoring, or any extensions you may need. The following partitions are created by default when initializing the Microsoft Azure image:
   - **100 GB** for the root partition
   - **500 GB** for spacewalk storage
   - **80 GB** for the database.
2. Once all configurations are set, review the settings.

3. Click "Create" to deploy your VM.
4. After the VM is deployed, you can access it via SSH.

Usage and Costs

Keep in mind that since this is a PAYG image, you will be billed according to your actual usage, including the number of systems you **manage** and **monitor** with this instance. It’ s essential to regularly track and review your usage to prevent unexpected costs and ensure alignment with your needs.

Initial setup

Procedure: Login and update the system

1. SSH into your Microsoft Azure instance.
2. Switch to the root user and update the system.

```
sudo -i
zypper refresh
zypper update
```

Network Configuration

Before initiating **yast2 susemanager_setup**, ensure the network is correctly configured for Microsoft Azure.

For SUSE Manager PAYG setup it is required, that a new Virtual Network and a new subnet is configured. It is impossible to setup the instance in an existing network. When you setup new clients which should be managed, take care that you put them into the private subnet which is configured with SUSE Manager PAYG.

If you want to manage systems in an already existing network, you must configure **peering**.

For more information, see Azure tutorial.

Procedure: Network Configuration

1. Ensure the network configuration aligns such that **hostname -f** yields the identical name as the reverse DNS lookup of the private IP address. For instance, when executing **nslookup 10.0.0.X**.

Insert the private IP with its Fully Qualified Domain Name (FQDN) into **/etc/hosts**. For example: **10.0.0.4 instancename.internal.cloudapp.net**

1. Edit **/etc/sysconfig/network/config** and append **internal.cloudapp.net** to **NETCONFIG_DNS_STATIC_SEARCHLIST**.
2. Execute **netconfig update**.
3. Subsequently, **hostname -f** should return the same FQDN as obtained from **nslookup 10.0.0.X**.

When adding new virtual machines (VMs), such as SUSE Linux Enterprise Server 15 SP5, it’ s crucial to ensure that they are configured within the same subnet as the SUSE Manager instance. This step is essential for successful network setup and integration.

Additionally, be mindful of the geographical restrictions associated with your plans. Make sure to utilize the appropriate Azure accounts which correspond to the specific geographic zones of your project.

Configure Storage

1. Configure storage with the **suma-storage** tool. For help see: **suma-storage --help** This tool simplifies creating the storage disk device and database device.

Use **lsblk** to see the names of your disk devices. You will see a list of partitions that were created when we configured the Microsoft Azure image.

- Root partition 100 GB
- Spacewalk storage 500 GB
- Database storage 80 GB

    1. Format the command in the following manner:

```
suma-storage <storage-disk-device> [<database-disk-device>]

#For example:

suma-storage /dev/nvme1n1 /dev/nvme2n1
```

This command will create the following storage locations:

~~~
``` /manager_storage /manager_storage/spacewalk ```

``` /var/spacewalk /var/cache/rhn /var/lib/spacewalk ```
~~~

Run setup

Execute **yast2 susemanager_setup** as root.

For more information on a typical SUSE Manager installation, see **Installation-and-upgrade › Server-setup**.

SUSE Manager PAYG WebUI setup

Requirements

SUSE Manager must be setup and running on Microsoft Azure. For more information on finding and setting up SUSE Manager, see **Specialized-guides › Public-cloud-guide**.

The instance-flavor-check tool and PAYG schedule check

instance-flavor-check Tool

The **instance-flavor-check** tool is an essential component that comes pre-installed by default. Its presence is crucial, as it determines the status of PAYG instances. When invoked, this tool identifies and categorizes the system as a PAYG instance. By default this tool is configured to run on a schedule named **update-payg-default**.

update-payg-default schedule task

Navigate to the **Admin › Task Schedules** to find the **update-payg-default** task. By default, this task is set to run at 10-minute intervals. Its primary function is to monitor the presence of all running PAYG instances by executing the **instance-flavor-check tool**.

Login to the Webui

Procedure: Login

1. Login to the SUSE Manager Server Web UI. The username is **admin** while the password is the name of the instance with **-suma** appended. When the name of the instance is **suma-payg-mycompany**, the password is **suma-payg-mycompany-suma**.
2. Select **Admin › Setup Wizard**.

Configure the PAYG credentials and product list

On a new server, the organization credentials table is listed at **Admin › Setup Wizard › Organization Credentials**. The Products table listed under **Admin › Setup Wizard › Products** will be empty.

Procedure: Verify credentials for PAYG and refreshing the product list

1. On your new server navigate to the **Admin › Setup Wizard › Pay-as-you-go** tab to view and manage PAYG credentials. Localhost should provide the description **SUSE Manager Pay-as-you-go** and the status should read **Credentials successfully updated**.
2. Navigate to the **Admin › Products** tab and click **[Refresh]**.
3. Once the refresh completes the following products are listed:
   - **SUSE Manager Proxy**
   - **SUSE Manager Server**
   - **Free products**

Ubuntu and Debian

Ubuntu and Debian are not currently supported by our RMT servers.

Red Hat Linux Enterprise Clients

SUSE Manager offers support for Red Hat Enterprise Linux clients; however it is important to note that content still needs to be added manually and is not available through the cloud-based RMT service.

For more information, see **Client-configuration › Registration-overview-redhat**.

PAYG Client connections and Client Registration

Managing connections and registering PAYG and BYOS images

PAYG (Pay-As-You-Go) Images

Connecting and registering PAYG images is supported and can be done without additional prerequisites.

BYOS (Bring Your Own Subscription) Images

Registering BYOS images requires SCC (SUSE Customer Center) credentials beforehand. If a user attempts to onboard a BYOS image without providing SCC credentials, the operation will fail, as these credentials are required to verify your subscription.

- Always verify the type of image (PAYG or BYOS) prior to initiating the registration process with SUSE Manager to ensure a smooth and error-free connection.
- If you have SCC credentials ensure your SCC credentials are valid and have the necessary permissions for utilizing BYOS images.

Do not register PAYG instances with the SUSE Customer Center. Only BYOS instances should be registered against the SUSE Customer Center.

Registering PAYG instances with the SUSE Customer Center or your own RMT server will create conflicts that are not easily solved. PAYG instances are automatically registered against the correct update server when added to your cloud service provider SUSE Manager instance.

Starting up additional PAYG SUSE products in Microsoft Azure

For more information on using SUSE PAYG products such as SUSE Linux Enterprise in the public cloud see: SUSE Linux Enterprise Public Cloud Guide

Establishing a PAYG SUSE Linux Enterprise connection with SUSE Manager PAYG

It is important to recognize that there is a difference between **Pay-as-you-go connections** and **bootstrapping**. Bootstrapping is also known as **client registration**.

Once you have started up your SUSE Manager PAYG instance and refreshed products you can add **Pay-as-you-go connections** via the **Admin › Setup Wizard › Pay-as-you-go** [**+Add Pay-as-you-go**] button. This action links your SUSE Manager PAYG instance with additional PAYG products on Microsoft Azure.

Bootstrapping or **client registration** is the process of onboarding clients in SUSE Manager. The registration process installs the appropriate client tools and provides full management functionality.

To manage your clients, start by **establishing connections** and then proceed to

bootstrap your clients. These two actions are the steps required for initiating and completing the client onboarding process.

Procedure: Connecting SUSE Linux Enterprise PAYG instances to SUSE Manager PAYG

1. Return to the SUSE Manager Server WebUI.
2. Navigate to **Admin › Setup Wizard › Pay-as-you-go** and click the [**+Add Pay-as-you-go**] button.
3. Add a description.
4. Complete the instance SSH connection data:
   - Host: **CSP URL of the machine**
   - SSH Port: **22**
   - User: **root**
   - Password: **<password>**
   - SSH Private Key
5. Click [**Create**] and wait for the sync to finish.
6. Proceed to **Admin › Setup Wizard › Products** to access your PAYG SUSE Linux Enterprise products.

Registering Clients

Now that you have established connections to the SUSE Manager PAYG instance you can register them as clients. To learn how to register various clients see: **Client-configuration › Registration-methods**

1.3. SUSE Manager PAYG on the Cloud: FAQ

Last updated: October 2023

Introduction

To support customers that prefer a cloud centric billing model, SUSE has recently published a new listing to the AWS & Azure Marketplaces, SUSE Manager

This brings the value of running SUSE Manager to cloud customers, with the benefit of a new pay-monthly pricing model via the cloud.

Documents in this FAQ

- **Specialized-guides › Public-cloud-guide**
- **Specialized-guides › Public-cloud-guide**
- **Specialized-guides › Public-cloud-guide**
- **Specialized-guides › Public-cloud-guide**

- **Specialized-guides › Public-cloud-guide**
- **Specialized-guides › Public-cloud-guide**

Public Cloud Listings

Last updated: October 2023

What is the SUSE Manager PAYG listing on the cloud marketplaces?

By selecting the SUSE Manager PAYG listing from the cloud marketplace, customers can deploy SUSE Manager to their cloud environment with the advantage of having monthly billing via the cloud provider.

Customers still have the option of using SUSE Manager byos (Bring-your-own-subscription), for which subscriptions and billing is handled directly by SUSE.

Where do I find the SUSE Manager PAYG listings?

The listings can be found in the AWS & Azure Marketplaces, there are two listings:

On the AWS Marketplace, you will find the following:

- SUSE Manager Server Family (Non-EU and non-UK only)
- SUSE Manager Server Family (EU and UK only)

On the Azure Marketplace, you will find:

- SUSE Manager with 24x7 Support
- SUSE Manager with 24x7 Support (EMEA Orders Only)

For the purposes of this FAQ, we will refer to these listings as SUSE Manager PAYG.

Why are there 2 listings and which one should I use?

We have 2 listings for SUSE Manager, "EU and UK only / EMEA Orders Only" and "non-EU and non-UK only", you should pick the listing that reflects in which country your AWS & Azure account gets billed.

Are these listings available in all countries?

The SUSE Manager PAYG listing on the cloud is not available to purchase in all countries. In case it is not available in your country you can still use the BYOS option.

Your billing country is based on the AWS & Azure Account ID used to do the deployment.

You can find out more about confirming your tax location as follows:

- For Amazon Web Services, it depends on the **tax address** of the account, see Tax help based on location
- For Microsoft Azure, it depends on the **sold to** address of the account, see Azure account profile

Please see the link at the end of this FAQ for a list of countries that can and cannot transact SUSE Manager PAYG via the cloud marketplaces.

My cloud account is in the USA, but I want to deploy SUSE Manager PAYG in another region, a region that is in a country where I currently cannot transact, is this possible?

If your cloud provider account is billed to one of the allowed countries, it is possible to deploy SUSE Manager in any region.

Is this listing available in China?

It is not possible to transact/bill SUSE Manager PAYG via AWS & Azure Marketplaces in China. It is possible to deploy into regions in China.

This means you can deploy SUSE Manager in a region in China if your billing is performed in a country that has the PAYG option available. Please see the addendum at the end of this FAQ for a list of countries that can and cannot transact SUSE Manager PAYG via the cloud.

Is there still a BYOS listing for SUSE Manager in the marketplace?

Yes, for customers that wish to use an existing annual SUSE Manager subscription, the BYOS SUSE Manager listing is available.

Is there a PAYG listing available for the SUSE Manager Proxy Server?

If a SUSE Manager Proxy is required as part of the architecture, it is suggested to use the containerized proxy. See the SUSE documentation at: documentation.suse.com **Select › SUSE Manager › SUSE Manager {4.3} › Installation and Upgrade guide › Installation › SUSE Manager Proxy › Install Containerized Proxy**

From the subscription point of view, SUSE Manager Proxy is a managed client.

The listing shows pricing information for SUSE Manager node management and a separate price for monitoring. Do I need to subscribe twice if I need monitoring?

There is no need to subscribe twice. Subscribing to the listing in the cloud provider marketplace allow the SUSE Manager PAYG instance to perform metered billing for SUSE Manager usage.

If you onboard an instance and enable SUSE Manager monitoring, there is a monthly charge for management and another charge for monitoring.

Billing

Billing - General

Last updated: October 2023

I have an existing SUSE Manager BYOS subscription; can I use this on the cloud?

BYOS Bring-your-own-subscription SUSE Manager deployments are supported on the cloud, however, billing will not be via the cloud marketplace.

Once the existing subscription term ends, you can purchase SUSE Manager PAYG via the cloud provider marketplace and manage your instances with a SUSE Manager deployment that is billed via the cloud.

I have an existing deployment covered by a BYOS SUSE Manager subscription; can I use this new listing in the cloud for a separate deployment of SUSE Manager PAYG?

Yes, the listing works independently from your existing BYOS SUSE Manager subscriptions. Please remember that support processes may be different for deployments using your existing BYOS subscription and those billed via the cloud and note that there is currently no process for migrating deployments between BYOS and PAYG versions.

Tell me more about how the billing for SUSE Manager PAYG works via the cloud?

When purchasing SUSE Manager via the AWS & Azure Marketplaces, the billing is as follows:

- Billing is monthly and handled via cloud marketplace.
- Managed instances are recorded hourly.
- The peak usage total is recorded for the month.
- There is a monthly usage charge for each instance in the count.

I have existing SLES for SAP instances procured via the cloud provider marketplace, is there a cost to manage these instance with SUSE Manager PAYG?

SLES for SAP Applications PAYG/on-demand instances, when purchased via the cloud marketplace, include Lifecycle Management (LCM) entitlements for SUSE Manager.

The SUSE Manager PAYG billing engine recognizes the embedded LCM entitlements and does not include them in the count. This means there are no additional charges to manage these with the SUSE Manager PAYG listing.

However, if SUSE Manager Monitoring is required for these instances, additional charges will apply for this feature.

Is there a way to try SUSE Manager PAYG before purchasing?

If using the SUSE Manager PAYG listing in the cloud marketplace, billing will commence from the time of deployment and when applicable client instances are on-boarded to SUSE Manager.

There is the SUSE Manager BYOS listing in the cloud marketplace that can be used, a trial subscription key can be obtained via the SUSE Website. This deployment cannot be directly converted to allow billing via cloud marketplaces.

To benefit from monthly billing via the cloud marketplace, SUSE Manager needs to be redeployed using the version from the PAYG listing in the cloud marketplace.

How does SUSE calculate the amount of managed instances to bill for?

Billing is based on the number of managed instances.

Below are 3 examples of how the node count is calculated, check the table below for the detail.

Static usage

Using SUSE Manager with 10 instances registered for management, for 1 month with no additional managed clients added in the month.

Bursting Model

Using SUSE Manager with 10 instances registered for management for 3 weeks in the month, bursting to 30 managed instances for 1 week.

Transient model

A temporary deployment of SUSE Manager with 20 managed instances, for 2 weeks (336 hours)

The total is calculated by counting number of active nodes (checked hourly). There is a monthly charge for each managed node under management in the billing cycle.

Below are 3 examples of how the average node count is calculated.

表格 3. Usage calculations for different scenarios

	Managed Instances	Peak Usage	Billed for
Static Usage	10	10	10
Bursting Model	10 - 3 Weeks, 30 - 1 Week	30	30
Transient Cluster	20 - 2 Weeks	20	20

Are special commercial terms available?

It is dependent on the deployment and the cloud provider, as every Cloud Service Provider (CSP) has its own commercial terms and offers.

It may be possible to secure special commercial terms such as an annual subscription. For example, AWS can offer SUSE Manager PAYG via a Private Offer. Please contact SUSE for more information.

Can my spending on SUSE Manager PAYG count towards my committed spend with the cloud provider?

There are such programs at the cloud providers:

- Azure MACC Program
- AWS Enterprise Discount Program
- Google CUD

Yes, it can. Please contact your cloud provider sales team for more details.

How do I purchase SUSE Manager / LCM subscriptions for additional managed instances?

Once SUSE Manager PAYG has been deployed from the marketplace listing and billing is active, there is no need to make a specific purchase to manage additional instances. Billing is dynamic and based on the number of instances SUSE Manager is managing. Just on-board additional instances to SUSE

Manager as needed, this will be reflected in your cloud marketplace bill.

Is this an annual commitment, will it auto-renew?

By default, the SUSE Manager PAYG listing on the cloud is billed on a monthly cycle, based on usage. Billing is ongoing for as long as SUSE Manager is deployed and instances are registered.

Is there a volume discount / tiered pricing built into the Cloud Listing?

Currently, there is no volume discount available.

I no longer require support; how can I end the SUSE Manager PAYG subscription?

If you no longer require support, there are 2 options:

- Unregister all systems from the SUSE Manager Server
- Terminate the SUSE Manager instance.

Billing - Technical

Last updated: October 2023

Do I need any special or additional infrastructure to set up SUSE Manager billing via the cloud marketplaces?

No, all components needed to handle the billing are part of the SUSE Manager PAYG code base.

What is the deployment mechanism for SUSE Manager PAYG on the cloud marketplace?

The cloud marketplace listing for SUSE Manager PAYG will deploy a SUSE Manager instance, together with the embedded billing engine.

What version of SUSE Manager is installed when using the marketplace listing?

The cloud marketplace listing for SUSE Manager PAYG is tied to a specific version of SUSE Manager (4.3.8 at launch). Future listings will typically include the latest version available at the time of product/image creation. Please check the listing for further information.

I need a prior version of SUSE Manager installed; can I still use the listing?

No, there is no choice over the SUSE Manager version when deploying using the cloud marketplace listing. If a prior version of SUSE Manager is required, this will need to be installed manually using the standard documentation and BYOS billing.

How often is the listing updated (including the version of SUSE Manager etc.)?

The Marketplace listing is tied to a specific version of SUSE Manager, which is the latest version available at the time of the listing.

Typically, the listing in the cloud marketplace is updated quarterly, or more frequently if there are security issues.

It is worth noting that the image update process may not be fully aligned to the SUSE Manager product maintenance releases (every 6-8 weeks), so SUSE Manager PAYG instances should be patched regularly after deployment.

I have purchased multiple PAYG - Pay-as-you-go SUSE products from the cloud provider marketplace (i.e. Rancher, NeuVector and SUSE Manager), does the marketplace billing method still work?

Yes, the billing mechanism for the products is independent. Each deployment will be billed separately via the cloud marketplace.

I already have an existing SUSE Manager deployment; how can I get this deployment to bill via the cloud marketplace?

Existing deployments **cannot** be directly converted or updated to provide billing via the cloud marketplace. To benefit from monthly billing, SUSE Manager PAYG must be deployed directly from the cloud marketplace listing.

Billing - Products

How do I migrate an existing SUSE Manager BYOS instance to the PAYG - Pay-as-you-go version?

Currently, it is **not possible** to migrate an existing instance to the PAYG version.

Customers can simply deploy a new SUSE Manager PAYG instance from the cloud marketplace and re-onboard all instances using the standard process.

Client configuration overview

How do I get support?

It is simple to open a support case with SUSE for SUSE Manager PAYG.

Create a ‘supportconfig’ via SUSE Manager PAYG instance CLI and upload the output to the SUSE Customer Center (SCC) at the following link:

https://scc.suse.com/cloudsupport

Providing marketplace billing is active, a support case will be opened.

Are there any differences between the SUSE Manager PAYG product on the cloud marketplaces compared to the versions I can run in my own data center?

The SUSE Manager PAYG product available in the cloud marketplace is the same product, with similar functionality that you would run on-premises or with a manual installation.

The only exceptions being the ability to bill via the cloud marketplace and connect to the SUSE Update Infrastructure for SUSE Manager product updates.

It is also worth checking the list of supported client operating systems to ensure your needs are met.

Does the SUSE Manager PAYG instance need to run 24/7?

It is recommended that the SUSE Manager instance always remains active.

How do I get fixes and updates for SUSE Manager PAYG, do I need a subscription key?

SUSE Manager PAYG, when deployed via the cloud marketplace will automatically connect to the SUSE Update Infrastructure, no separate SUSE Manager subscription key is required.

Updates to the SUSE Manager instance can be applied from those channels using standard SUSE tooling (i.e. zypper).

Is the use of SUSE Manager Hub supported with this listing?

Yes, SUSE Manager Hub is recommended for customers with more than 10,000 managed clients. If the SUSE Manager Hub instance has no connected clients, there will be no charge for the SUSE Manager Hub software components.

Billing will occur from the downstream SUSE Manager PAYG Server instance, where managed instances are connected.

Can I manage Bring-your-own-subscription - BYOS SLES instances in the cloud with SUSE Manager PAYG?

To manage BYOS instances, SUSE Customer Center (SCC) credentials with entitlement to valid SLES or SLES for SAP subscriptions will be required. These credentials should be added to the SUSE Manager configuration to enable this functionality.

I have a hybrid setup; can I manage instances (workloads) outside of the cloud?

Yes, it is possible to manage instances, in the cloud, on premises or both.

To enable management of SUSE workloads in an on-premises data center, SCC credentials with entitlement to valid SLES or SLES for SAP subscriptions will be required.

These credentials should be added to the SUSE Manager configuration to enable this functionality.

```
Egress data charges may apply when managing workloads outside of the
cloud provider's network.
```

Are there additional charges for using a SUSE Manager Proxy?

With a hybrid landscape, egress data charges may apply when managing workloads outside of the cloud provider. To reduce these costs, SUSE recommend the use of the containerized proxy.

Please note: Whilst the SUSE Manger Proxy usage is included as part of the SUSE Manager PAYG subscription, an LCM+ subscription will be billed if the proxy is also a managed workload, so in other words it counted as a managed client.

Can I have a SUSE Manager PAYG Server on premises and bill via the Cloud Marketplace.

No, to benefit from cloud marketplace billing, the SUSE Manager PAYG instance must reside on the cloud. Any SUSE Manager instances on premises will require their own subscriptions.

What client Operating Systems are supported with SUSE Manager PAYG?

Please refer to the product documentation for a full list of supported clients. PAYG Overview

Miscellaneous

Where can I find out more about the Cloud Marketplace listing for SUSE Manager?

You can use the following resources to learn more about the SUSE Manager listing and how to deploy.

- New dawn for SUSE Manager
- documentation.suse.com/suma/

Where can I find out more about SUSE Manager?

- SUSE Manager
- Training courses

Appendix

Countries that can transact SUSE Manager PAYG through the cloud marketplace

Countries that can transact via the cloud marketplaces: Supported Country List

Public Cloud Support

BYOS and PAYG Support

Regardless of the public cloud service you use or the plan you choose, SUSE has you covered with support. Where you get support is dependent on whether you **Bring-your-own-subscription** (BYOS) or you use SUSE products **on-demand** (PAYG).

BYOS

BYOS instances are supported by SUSE under the terms of your SUSE subscription. For an overview of SUSE’ s support subscriptions, refer to https://www.suse.com/support/..

PAYG

SUSE Manager PAYG instances include support by SUSE. To open a ticket for PAYG, see https://scc.suse.com/cloudsupport.

Public Cloud Guide

For more information on cloud products and support, see https://documentation.suse.com/sle-public-cloud/all/html/public-cloud/public-cloud.html.

Public Cloud Information Tracker (PINT)

The Public Cloud Information Tracker (PINT) provides information about the images SUSE publishes and servers that are part of the SUSE operated update infrastructure. PINT is available at https://pint.suse.com and provided as an API and command-line tool with the **python3-susepubliccloudinfo** package from the Public Cloud Module repository.

Collecting System Information with supportconfig

To create a TAR archive with detailed system information that you can hand over to Global Technical Support, use the command **supportconfig**. The command-line tool is provided by the package **supportutils** which is installed by default.

Depending on which packages are installed on your system, some of these packages integrate **supportconfig** plug-ins. When **supportconfig** is executed, all plug-ins are executed as well, creating one or more result files for the archive. This has the benefit that the only topics checked are those that contain a specific plug-in for them. **supportconfig** plug-ins are stored in directory **/usr/lib/supportconfig/plugins/**.

The following procedure shows how to create a **supportconfig** archive, but without submitting it to support directly.

Procedure: Creating a supportconfig Archive

1. Run **supportconfig** as root. Usually, it is enough to run this tool without any options. Some options are very common and are displayed in the following list:

   #### 列表 1. Options:

   ```
   -E MAIL, -N NAME, -O COMPANY, -P PHONE

   Sets your contact data: e-mail address (-E)
   Company name (-O)
   Your name (-N)
   Your phone number (-P)

   -i KEYWORDS, -F
   Limits the features to check.
   The placeholder KEYWORDS is a comma-separated list of case-sensitive keywords.
   Get a list of all keywords with supportconfig -F.
   ```

2. Wait for the tool to complete the operation.
3. The default archive location is **/var/log**, with the file name format being **scc_HOST_DATE_TIME.txz**

Opening a New Support Case

To open a new support case for your cloud Pay-as-you-go instance, you need to generate a **supportconfig** file and then follow the steps listed on:

https://scc.suse.com/cloudsupport

Billing Services and Troubleshooting

billing-data-service

This service serves billing data from SUSE Manager for the **CSP-billing-adapter**. The service is reachable at:

- **http://localhost:18888/**
- **http://localhost:18888/metering**

This service is maintained by the SUSE Manager team. This service only logs to **systemd journal**.

Billing services must be running or **spacewalk** will refuse to start.

csp-billing-adapter-service

This service reports the billing data to the cloud provider (AWS, AZURE, GCP). These logs are stored in:

- **/var/log/csp_billing_adapter.log**

Additional status information can be located in:

- **/var/lib/csp-billing-adapter/csp-config.json**

This service is maintained by public cloud team.

mgr-check-payg.service

This service checks if **billing-data-service** is installed and running. It also checks if the **csp-billing-adapter-service** is installed and running. If these services are not running or installed it will attempt to install and start the service.

Consequences

If one of the previous billing services is not running and cannot be installed or started **spacewalk** will refuse to start.

Check **journalctl -u mgr-check-payg.service** for additional issues.

Continuous Compliance Checks

Procedure: Compliance Checks

1. The **payg-dimension-computation** taskomatic task must be active.
2. Is the billing server running at **http://localhost:18888/** ?
3. Is the **csp-billing-adapter.service** running?
4. What is the status of the **csp-billing-adapter** located in **/var/lib/csp-billing-adapter/csp-config.json**?

The following files and packages are checked for tampering

- Billing-data-service
- Csp-billing-adapter-service
- Python3-csp-billing-adapter
- Python3-csp-billing-adapter-local
- Python3-csp-billing-adapter-amazon
- python3-csp-billing-adapter-azure

Failed Checks

Failed checks are logged in:

- **/var/log/rhn/rhn_web_ui.log**
- **/var/log/hrn/rhn_taskomatic_daemon.log**

Consequences

- No more actions will be executed.
- **reposync** will stop synchronizing channels.

Salt Guide Overview

Updated: 2025-07-21

Salt is a remote execution engine, configuration management and orchestration system used by SUSE Manager to manage clients.

In SUSE Manager, the Salt master runs on the SUSE Manager Server, allowing you to register and manage Salt clients.

This book is designed to be a primer for using Salt with SUSE Manager.

For more information about Salt, see the Salt documentation at https://docs.saltproject.io/en/latest/contents.html.

The current version of Salt in SUSE Manager is 3006.0.

> Throughout the SUSE Manager documentation, we use the term **Salt clients** to refer to Salt machines that are connected to and controlled by the Salt master on the SUSE Manager Server. This is to clearly differentiate them from traditional clients. In other documentation, and in some internal references, Salt clients are sometimes referred to as Salt **minions** instead. This is a difference in terminology only.

Terminology

Beacon

Beacons allow you to use the Salt event system to monitor non-Salt processes. Clients can use beacons to connect to various system processes for constant monitoring. When a monitored activity occurs, an event is sent on the Salt event bus that can then trigger a reactor.

> To use beacons on SUSE Linux Enterprise Server Salt clients, install the **python-pyinotify** package. For Red Hat Enterprise Linux systems, install the **python-inotify** package.

For more information on beacons, see https://docs.saltproject.io/en/latest/topics/beacons/

Broker

The Salt broker allows clients to pass commands to each other. The broker acts like a switch, therefore peer communication will only work for clients on the same network, or connected to the same proxy.

For more information on Salt and peer communication, see https://docs.saltproject.io/en/latest/ref/peer.html.

Environment

SUSE Manager implements Salt with a single environment. Multiple Salt environments are not supported.

Formulas

Formulas are collections of Salt States that contain generic parameter fields. Formulas are used within SUSE Manager to assist with configuring Salt clients. Some formulas have extensive configuration options, and use forms to help organize them in the SUSE Manager Web UI.

For more information about formulas, see **Specialized-guides › Salt**.

Grains

Grains provide information about the hardware of a client. This includes the operating system, IP addresses, network interfaces, and memory. When you run a Salt command any modules and functions are run locally from the system being called.

For more information on grains, see https://docs.saltproject.io/en/latest/topics/grains/.

Highstate

This term is used when you apply all outstanding states to all targeted clients at the same time. The highstate must be applied when doing changes to systems, including enabling and disabling formulas.

Key Fingerprints

Key fingerprints are exchanged between the SUSE Manager Server and Salt clients to verify the identity of the server and the client. This prevents Salt clients from connecting to the wrong server. You can see the fingerprints of your Salt clients by navigating to **Salt › Keys**.

Master

The Salt master issues commands to its attached clients. In SUSE Manager, the Salt master must be the SUSE Manager Server.

Minions

Salt clients that are connected to and controlled by the Salt master on the SUSE Manager Server. In SUSE Manager, these are referred to as Salt clients, in order to clearly differentiate them from traditional clients. This is a difference in terminology only.

Modules

Functions within Salt are stored in modules. Salt modules are stored on clients and the SUSE Manager Server within the **/usr/lib/python*/site-packages/salt/** directory. There are many types of Salt modules, including state and execution modules. You can write your own Salt modules using Python.

For a complete list of available Salt modules, see https://docs.saltproject.io/en/latest/ref/index.html.

Pillars

Pillars are created on the SUSE Manager Server. They contain information about a client or group of clients. Pillars allow you to send confidential information to a targeted client or group of clients. Pillars are useful for sensitive data, configuration of clients, variables, and any arbitrary data.

For more information on pillars, see https://docs.saltproject.io/en/latest/topics/tutorials/pillar.html.

States

States are configuration templates. They allow you to describe what each of your systems should look like, including the applications and services that are installed and running. States are applied to the target client. This automates the process of bringing a large number of systems into a known state, and then maintaining them.

> Do not update the **salt** package using states. Update all other system packages using states. You can then update the **salt** package from the SUSE Manager Web UI as a separate step.

For more information on states, see https://docs.saltproject.io/en/latest/topics/tutorials/starting_states.html.

For more Salt terminology, see https://docs.saltproject.io/en/latest/glossary.html.

Salt Command

Salt commands have three main components: target, function, and arguments. The calls are constructed in this format:

```
salt 'target' <function> [arguments]
```

The target defines the client, or group of clients, on which to run the function.

The function is the particular task to be run.

Arguments provide any extra data required by the function.

Salt Targets

Salt command targets allow you to specify a client or group of clients. There are several different targets you can use.

General Targeting

List available grains on all clients:

```
salt '*' grains.ls
```

Target a specific client:

```
salt 'web1.example.com' test.ping
```

Glob Targeting

Target all clients using a particular domain:

```
salt '*example.com' test.ping
```

Target all clients using a particular label:

```
salt 'label*' test.ping
```

List Targeting

Specify a flat list of clients, using their IDs:

```
salt -L 'client_ID1, client_ID2, client_ID3' test.ping
```

Regular Expression Targeting

You can also define targets with PCRE-compliant regular expressions:

```
salt -E '(?!web)' test.ping
```

IP Address Targeting

List available client IP addresses:

```
salt '*' network.ip_addrs
```

Target a specific client IP address:

```
salt -S '172.31.60.74' test.ping
```

Target all clients on a subnet:

```
salt -S 172.31.0.0/16 test.ping
```

For more on targeting, see https://docs.saltproject.io/en/latest/topics/targeting/.

Salt Execution Modules

When you have specified a target, provide the module and function to execute on the target.

Find which modules can be executed on the target:

```
salt '*' sys.doc
```

For a full list of callable modules, see https://docs.saltproject.io/en/latest/ref/modules/all/index.html.

Salt Function Arguments

Functions accept arguments for any extra data.

For example, the **pkg.install** function requires an argument specifying which package to install:

```
salt '*' pkg.install yast2
```

You can provide more than one argument to a function, with spaces between them. For example:

```
salt '*' cmd.run 'echo "Hello: $FIRST_NAME"' env='{FIRST_NAME: "John"}'
```

Often Used Salt Commands

This section contains the most commonly used Salt commands. For a complete list of available Salt commands, see https://docs.saltproject.io/en/latest/ref/cli/index.html.

salt-run

Display all clients that are running:

```
salt-run manage.up
```

Display all clients that are not running:

```
salt-run manage.down
```

Display the current status of all Salt clients:

```
salt-run manage.status
```

Check the version of Salt running on the SUSE Manager Server and active clients:

```
salt-run manage.versions
```

salt-cp

Copy a file to a client or set of clients:

```
salt-cp '*' foo.conf /root
```

salt-key -l

List public keys:

```
salt-key -l all
```

salt-key -a my-minion

Accept pending key for a minion:

```
salt-key -a my-minion
```

salt-key -A

Accept all pending keys:

```
salt-key -A
```

salt grains

List all available grains:

```
salt '*' grains.ls
```

List collected grain system data:

```
salt '*' grains.items
```

Salt States and Pillars

States are configuration templates. They allow you to describe what each of your systems should look like, including the applications and services that are installed and running. Salt state files are referred to as SLS (SaLt State) files.

States are applied to the target systems by matching relevant state data to clients. The state data comes from SUSE Manager in the form of package and custom states.

You can target clients at three specific levels of hierarchy and priority: individual clients, system groups, and organization. Individual clients have priority over groups, and groups have priority over the organization.

For example:

- The Organization requires that version 1 is installed. All clients are part of the same Organization.
- Group A requires that version 2 is installed. Client1, Client2, and Client3 are part of Group A.
- Group B requires any version installed. Client4 is part of Group B.

Leading to these possible scenarios:

- Client1 wants package removed, package is removed (Client Level)
- Client2 wants version 2, gets version 2 (Client Level)
- Client3 wants any version, gets version 2 (Group Level)

- Client4 wants any version, gets version 1 (Organization Level)

For more information on Salt states, see https://docs.saltproject.io/en/latest/topics/states/.

You can create custom Salt states with SUSE Manager. For more information, see **Specialized-guides › Salt**.

Group States

Pillar data can be used to perform bulk actions, like applying all assigned states to clients within the group. This section contains some examples of bulk actions that you can take using group states.

To perform these actions, you will need to determine the ID of the group that you want to manipulate. You can determine the Group ID by using the **spacecmd** command:

```
spacecmd group_details
```

These examples use an example Group ID of **GID**.

To apply all states assigned to the group:

```
salt -I 'group_ids:GID' state.apply custom.group_GID
```

To apply any state (whether or not it is assigned to the group):

```
salt -I 'group_ids:GID' state.apply ``state``
```

To apply a custom state:

```
salt -I 'group_ids:2130' state.apply manager_org_1.``customstate``
```

Apply the highstate to all clients in the group:

```
salt -I 'group_ids:GID' state.apply
```

Salt Pillars

SUSE Manager exposes a small amount of internal data as pillars which can be used with custom states. Pillars are created on the SUSE Manager Server, and contain information about a client or group of clients. For custom information in pillars, see **Client-configuration › Custom-info**. Pillars are useful for sensitive data, configuration of clients, variables, and any arbitrary data.

Pillars are managed either automatically by SUSE Manager, or manually by the user.

If you change pillar data on the server (Salt master) the actual pillar data on the client (minion) is updated only after calling **saltutil.refresh_pillar** for the client.

Otherwise it could happen that **pillar.items** and **pillar.get** calls would produce different results with different values of pillars that were not refreshed.

To avoid hard-coding organization IDs within SUSE Linux Enterprise Server files, a pillar entry is added for each organization:

```
org-files-dir: relative_path_to_files
```

The specified file is available for all clients which belong to the organization.

This is an example of a pillar located at **/etc/motd**:

```
file.managed:
    - source: salt://{{ pillar['org-files-dir']}}/motd
    - user: root
    - group: root
    - mode: 644
```

For more information on Salt pillars, see https://docs.saltproject.io/en/latest/topics/pillar/.

Download Endpoint

By default, SUSE Manager assumes that the download endpoint to use is the FQDN of the SUSE Manager Server or Proxy. However, there are some cases where you might like to use a different FQDN as the download endpoint. The most common example is if you need to use load balancing, caching proxies, or in environments with complicated networking requirements.

To change the package download endpoint, you can manually adjust three Salt pillars: * **pkg_download_point_protocol**, defaults to **https**. * **pkg_download_point_host**, defaults to the FQDN of the SUSE Manager Server (or Proxy, if in use). * **pkg_download_point_port**, defaults to **443**.

If you do not adjust these pillars directly, SUSE Manager will fall back to the default values.

Procedure: Changing the Package Download Endpoint Pillar

1. Navigate to **/srv/pillar/** and create a file called **top.sls** with these contents:

   ```
   base:
     '*':
       - pkg_download_points
   ```

   This example directs Salt to look at the **pkg_download_points.sls** file to determine the base URL to use. You can adjust this file to target different clients or groups, depending on your environment.

2. Remain in **/srv/pillar/** and create a file called **pkg_download_points.sls** with the base URLs you want to use. For example:

   ```
   pkg_download_point_protocol: http
   pkg_download_point_host: example.com
   ```

```
pkg_download_point_port: 444
```

3. OPTIONAL: If you want to use external pillars, for example Group IDs, open the master configuration file and set the **ext_pillar_first** parameter to **true**. You can then use Group IDs to set conditional values, for example:

```
{% if pillar['group_ids'] is defined and 8 in pillar['group_ids'] %}
  pkg_download_point_protocol: http
  pkg_download_point_host: example.com
  pkg_download_point_port: 444
{% else %}
  pkg_download_point_protocol: ftp
  pkg_download_point_host: example.com
  pkg_download_point_port: 445
{%- endif %}
```

4. OPTIONAL: You can also use grains to set conditional values, for example:

```
{% if grains['fqdn'] == 'client1.example.com' %}
    pkg_download_point: example1.com
{% elif grains['fqdn'] == 'client2.example.com'' %}
    pkg_download_point: example2.com
{% else %}
    pkg_download_point: example.com
{% endif %}
```

GPG Encrypted Pillars

Salt has support to transparently decrypt GPG-encrypted Pillar data built-in. The decryption happens on the Salt Master.

Generate GPG keyring for Salt Master

The GPG keyring can be specified in **/etc/salt/master** or in its own file under **/etc/salt/master.d/**, for example **/etc/salt/master.d/gpg-pillar.conf**.

Always create a separate keyring for the Salt Master.

Procedure: Generating key pair

1. On the Salt Master create GPG home directory and restrict its permissions:

```
mkdir /etc/salt/gpgkeys
chmod 700 /etc/salt/gpgkeys
```

2. Generate a key pair interactively.

   The password must be empty.

```
gpg --gen-key --homedir /etc/salt/gpgkeys
```

3. Salt does not run with root permissions on SUSE Linux Enterprise and openSUSE distributions.

   ```
   chown -R salt:salt /etc/salt/gpgkeys
   ```

4. Configure Salt Master to use the new GPG home directory

   ```
   echo 'gpg_keydir: /etc/salt/gpgkeys' >/etc/salt/master.d/gpg-pillar.conf
   systemctl reload-or-restart salt-master
   ```

Use GPG for encrypting Pillar secrets

Salt GPG renderer decrypts GPG encrypted contents that are ASCI-armored. To use the GPG renderer in a Pillar YAML file, change

```
#!yaml
```

to

#!yaml|gpg

```
Encrypting pillar secrets can be done anywhere as long as the GPG and the public key
generated in <<proc-key-pair-generation>> are available.

In this example, "SUMA Salt Master" is the GPG key’ s UID created earlier.
```

echo 't0ps3cr3t' | gpg --armor --batch --encrypt --recipient "SUMA Salt Master"

```
When the GPG encrytped contents are created and available as ASCII-armored output, this
output can be used as a multi-line string in a pillar YAML file:
```

#!yaml|gpg

secret: my-secret: | -----BEGIN PGP MESSAGE-----

```
hQEMA3OrmRaWrqgqAQf/ej8xV+nO3HVbQRCeJgCmt5ZjnogT++HHeFzXymfr1SgT
XySyAqpIZB2N6MjZXtupO2sCmG6fzqtmnW+vRsZhQG8PAqzRtAekFuVbXzgkigBk
338yOdyltVBtMONnkHFQ+7EP1tfJnWLCVrJ1I42vGFLZf2AD1xhbjewCcoaK82J4
f8u9U/dxgV0N6na28WG5m6YU5Reu1Ca37PXHuqA/0XZl65DY63xaMPMDHZEi1wkU
GXU7OsiL1dO0/sST1Awo5i99kVt/kA6DCGDuxTNpLrauNLOKUbtwcxvavtNZGwdQ
yI9zWVx8qerWE0aO3M7zVDJftv77faV2ENiqzaadvtJHAZynW4GW7rSuP1RXFzlB
DOAmzdRuIJwiLC9R2BKu3x+avReQb6xoz7eF7WthC0H0dz4mYakwPlVZ5yqYa/+G
83i951rqAGI=
=g+ji
-----END PGP MESSAGE-----
```

```
When the pillar is assigned to a system with [path]``top.sls``, the GPG encrypted pillar
data is available in a decrypted format.


[NOTE]
```

```
====
The client's in-memory cache is only updated on startup or when running execution module
functions that trigger a cache refresh such as [literal]``saltutil.refresh_pillar``,
[litaral]``pillar.items``, or [literal]``state.apply``.
====
```

suma-sles15sp1.tf.local: ---------- my-secret: t0p s3cr3t!

```
== Export the GPG key

To export the GPG key, use the command:
```

gpg --export 'SUMA Salt Master' --homedir /etc/salt/gpgkeys --output suma-salt-master.gpg

```
Here 'SUMA Salt Master' is the name used during key generation.

The [literal]```suma-salt-master.gpg``` public key can be freely shared.

//For more information about GPG, see https://www.gnupg.org/documentation/.

:leveloffset!:
:leveloffset: +2

[[custom-states]]
= Custom Salt States

You can create your own custom Salt states with {productname} as centrally managed
configuration channels. Custom states are stored as Salt state files on the {productname}
Server with a ``.sls`` extension.



== Create a New Custom Salt Channel

You can use the {productname} {webui} to create and edit custom Salt state files. You must
create a state channel first, with an initial state named ``init.sls``. The ``init.sls``
file is used to reference all other state files within the channel. The custom states that
you create using the {webui} are stored on the {productname} Server in the the
[path]``/srv/susemanager/salt/<organization>/`` directory.

After the channel is created with an ``init.sls`` file, you can write additional state
files in the {webui}. Alternatively, you can upload existing state files to use within
your state channel, or import them from other channels or clients.



.Procedure: Creating a Custom Salt Channel and Initial State
. In the {productname} {webui}, navigate to menu:Configuration[Channels].
. Click btn:[Create State Channel].
. In the [guimenu]``Name`` field, type a name for your state.
. In the [guimenu]``Label`` field, type a label. Use alphanumeric characters, hyphens, and
underscores. Do not use spaces.
. In the [guimenu]``Description`` field, type a short description of the configuration
your state performs.
. In the [guimenu]``SLS Contents`` field, type the contents of your ``init.sls`` state. If
you want to reference file templates in this configuration channel, ensure your file
starts by specifying the source of the managed file, using this syntax:
+
```

file.managed: - source: salt://<org_name>/<channel_name>/etc/<ID>/<filename>

```
+
 Example custom state files are given later in this section.
. Click btn:[Update Channel] to save your state.


.Procedure: Adding Additional Files to a Custom State Channel
. In the {productname} {webui}, navigate to menu:Configuration[Channels].
. Click the name of the channel you want to add files to.
. To create a new file, click btn:``Create configuration file`` and type the contents of
the file.
. To upload an existing file, click btn:[Upload Configuration Files] and select the file
to upload.
. To copy an existing file, click btn:[Import a File from Another Channel or System] and
select the file to copy.



.Procedure: Editing a Custom Salt State
. In the {productname} {webui}, navigate to menu:Configuration[Channels].
. Click btn:[View/Edit <filename>.sls File].
. Make your changes to the file.
. Click btn:[Update Configuration File] to save your state.

You can also manage revisions, compare the state to others in your organization, and
download the ``.sls`` file from this dialog.



.Procedure: Assigning a Client to a Custom Salt State
. In the {productname} {webui}, navigate to menu:Configuration[Channels].
. Click the name of the state you want to assign a client to.
. Navigate to the menu:Systems[Target Systems] tab.
. Check the clients you want to assign.
. Click btn:[Subscribe systems].


For more information about Salt state modules, see
https://docs.saltproject.io/en/latest/ref/states/all/index.html.



== Example Custom State Files

This section contains some example custom state files. Use these as a basis for writing
your own custom states.



.Example: Manage a File
```

my_config_change_id: file.managed: - name: /etc/my.conf - source: salt://example_org/example_channel/etc/my.conf - user: root - group: root - mode: 644 - template: jinja

```
.Example: Package Management
```

my_pkg_id: pkg.installed: - refresh: True - pkgs: - glibc - kernel-default - hello: 1.0-42

```
.Example: Remote Command
```

ip_forward-on: cmd.run: - name: echo "1" > /proc/sys/net/ipv4/ip_forward - onlyif: - test **cat /proc/sys/net/ipv4/ip_forward** -eq 0

```
.Example: Service Management
```

time_service_id: service.running: - name: chronyd - enable: True

```
== Custom State to Trust a GPG Key

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

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

If you are bootstrapping Salt clients from the {productname} {webui}, you can use a custom
Salt state to trust the GPG key.



.Procedure: Trusting a GPG Key With a Custom Salt State
. Locate the key that you need to trust. Ensure you have the correct key, and that you
also have the fingerprint used to verify the key. This information is available from the
vendor or, in some cases, from a key server.
. Copy the key to a file location where the client can access it. We recommend saving it
in the [path]``/srv/www/htdocs/pub/`` directory, where all {suse} public keys are also
saved.
. In the {productname} {webui}, navigate to menu:Configuration[Channels].
. Click btn:[Create State Channel].
. In the [guimenu]``Name`` field, type a name for your state. For example, ``GPG Key
Trusts``.
. In the [guimenu]``Label`` field, type a label. For example, ``GPG_Key_Trusts``.
. In the [guimenu]``Description`` field, type a short description of the configuration
your state performs. For example, ``Trusts GPG Keys for CentOS``.
. In the [guimenu]``SLS Contents`` field, create a state to retrieve the appropriate key
from the {productname} Server and trust it on the client. The exact contents of your state
varies depending on your client operating system. For example:
+
```

rpm_trust_gpg_key: cmd.run: - name: rpm --import https://{{ salt['pillar.get']('mgr_server') }}/pub/<third-party-gpg>.key - unless: rpm -q gpg-pubkey-<key_id>

deb_trust_gpg_key: mgrcompat.module_run: - name: pkg.add_repo_key - path: https://{{ salt['pillar.get']('mgr_server') }}/pub/<third-party-gpg>.key

```
+
  Alternatively, you can add GPG keys to a configuration channel, using a managed file to
deploy them directly on the client.
  In this case, you would use a local path to the key, rather than a URL.
. Click btn:[Update Channel] to save your state.
. Navigate to menu:Configuration[Channels] and click the name of the state you want to
assign a client to.
. Navigate to the menu:Systems[Target Systems] tab and check the clients you want to
assign.
. Click btn:[Subscribe systems].
 When the configuration file is next run on the client, the GPG key is trusted.
```

```
Alternatively, you can manage your GPG keys from your own repository hosted on an external
file management system.



== Apply a custom state at highstate

To apply a custom state at highstate create a mapping in [path]``/srv/salt/top.sls``. This
short example maps the [literal]``test`` state to the system group [literal]``12``:
```

/srv/salt/top.sls

base: 'group_ids:12': - match: pillar - test

```
:leveloffset: 2
:leveloffset: +2

[[salt.file.locations]]
= Salt File Locations and Structure


There are several ways to set up the Salt file structure. This section describes how Salt
is supported and set up as part of {productname} Server. The main configuration file is
[path]``/etc/salt/master.d/susemanager.conf``.

[NOTE]
====
Do not edit the [path]``/etc/salt/master.d/susemanager.conf`` configuration file. This
file belongs to the [package]``spacewalk-setup`` package and is marked as
[literal]``%config``. When {suse} updates the [package]``spacewalk-setup`` package, the
[path]``susemanager.conf`` file is overwritten, and any customization is lost. Instead,
add your own configuration file to the [path]``/etc/salt/master.d/`` directory. This
prevents the update process from deleting your settings from the main
[path]``susemanager.conf`` configuration file.
====

Some settings from [path]``/etc/salt/master.d/susemanager.conf`` that can help with
finding configuration options:

[source]
```

Configure different file roots. Custom salt states should only be placed in

/srv/salt.

Users should not touch other directories listed here.

file_roots: base: - /usr/share/susemanager/salt - /usr/share/salt-formulas/states - /usr/share/susemanager/formulas/states - /srv/susemanager/salt - /srv/salt

Configure different pillar roots. Custom pillar data should only be placed

in /srv/pillar.

Users should not touch other directories listed here.

pillar_roots: base: - /srv/pillar

```
When you are working with [path]``/etc/salt/master.d/susemanager.conf``, be aware that:

* Files listed are searched in the order they appear
* The first matching file found is called

The {productname} Server reads Salt state data from five root directories:

[path]``/usr/share/susemanager/salt``::
This directory is shipped and updated with {productname} and includes certificate setup
and common state logic to be applied to packages and channels.

[WARNING]
====
Do not edit or add custom Salt data to this directory.
====

[path]``/usr/share/salt-formulas/states``::
[path]``/usr/share/susemanager/formulas/states``::
These directories are shipped and updated with {productname} or additional extensions.
They include states for Salt formulas.

[WARNING]
====
Do not edit or add custom Salt data to this directory.
====

[path]``/srv/susemanager/salt``::
This directory is generated by {productname}, based on assigned channels and packages for
clients, groups, and organizations. This directory will be overwritten and regenerated. It
is the Salt equivalent of the {productname} database.

[WARNING]
====
Do not edit or add custom Salt data to this directory.
====

Within this directory, each organization has a sub-directory.

.Example: SLS File Directory Structure
[source]
```

├── manager_org_<org id> │ ├── files │ │ … files needed by states (uploaded by users)… │ └── state.sls … other SLS files (created by users)… For example: ├── manager_org_TESTING │ ├── files │ │ └── motd # user created │ │ … other files needed by states … │ └── motd.sls # user created … other SLS files …

```
[path]``/srv/salt``::
This directory is used for custom state data, modules, and related data. {productname}
does not operate or use this directory directly. The state data in this directory is used
by the client highstate, and is merged with the total state result generated by
{productname}. Use this directory for custom Salt data.

The {productname} Server reads Salt pillar data from two root directories:
```

```
/usr/share/susemanager/pillar:: This directory is generated by {productname}. It is
shipped and updated together with {productname}.

[WARNING]
====
Do not edit or add custom Salt data to this directory.
====

/srv/pillar:: By default, {productname} does not operate or use this directory directly.
The custom pillar data in this directory is merged with the pillar result created by
{productname}. Use this directory for custom Salt pillar data.

[NOTE]
====
You can use the [systemitem]``gitfs`` fileserver backend to serve Salt data from git
repositories. For more information, see xref:specialized-guides:salt/salt-gitfs.adoc[].
====

:leveloffset: 2
:leveloffset: +2

[[salt.gitfs]]
= The gitfs Fileserver Backend

In {productname}, [package]``pygit2`` is the supported Python interface to git. When
[package]``pygit2`` is installed the ``gitfs`` fileserver backend is available and it is a
supported feature.


Configuration options are set in the [path]``/etc/salt/master`` file, or in a separate
configuration file in the [path]``/etc/salt/master.d/`` directory. The basic settings are:


fileserver_backend::
List of fileserver backends that the Salt master checks for files in the order they are
defined. Options:
+
* [literal]``roots``: Files local on the Salt master ({productname} Server).
    [literal]``roots`` is required to keep the product running. You can only enable
[literal]``gitfs`` optionally. Additionally, {suse} strongly recommends to prefer
[literal]``roots`` (local files) over [literal]``gitfs``. The standard backend.
* [literal]``gitfs``: Files stored in one or more git repositories.
    The repositories are defined with [literal]``gitfs_remotes``.
+
Example:
+
```

fileserver_backend: - roots - git

```
gitfs_remotes::
List of git repositories. ``git://``, ``https://``, ``file://``, or ``ssh://`` URLs can be
configured. For SSH remotes, a [command]``scp``-like syntax is also supported; for
example: [literal]``gitlab@gitlab.example.com:universe/setup.git``. Then you can also
specify options for credentials, file locations, or branches such as [literal]``pubkey``,
[literal]``privkey``, [literal]``root``,[literal]``base``.
+
Example:
+
```

gitfs_remotes: - https://example.com/myformulas/formula.git -

gitlab@gitlab.example.com:universe/setup.git: - pubkey: /var/lib/salt/.ssh/id_rsa_gitlab.pub - privkey: /var/lib/salt/.ssh/id_rsa_gitlab - root: srv/salt - base: master

```
ext_pillar::
List of external pillar interfaces. Salt can also serve pillar data from one or more git
repositories. For syntax and options, also see the [literal]``gitfs_remotes`` setting.
+
Example:
+
```

ext_pillar: - git: - master gitlab@gitlab.example.com:universe/setup.git: - root: srv/pillar - pubkey: /var/lib/salt/.ssh/id_rsa_gitlab.pub - privkey: /var/lib/salt/.ssh/id_rsa_gitlab

```
For more information, see:

* https://docs.saltproject.io/en/latest/topics/tutorials/gitfs.html
* https://docs.saltproject.io/en/latest/ref/configuration/master.html

:leveloffset: 2
:leveloffset: +2

[[yomi.installer]]
= Install Using Yomi

Yomi (yet one more installer) is an installer for {suse} and openSUSE operating systems.
Yomi is designed as a Salt state, and can be used for installing {suse} operating systems
on new systems.

In {productname}, Yomi can be used as part of provisioning new clients, as an alternative
to {ay}.

Yomi consists of two components:

* The Yomi formula, which contains the Salt states and modules required to perform the
installation.
* The operating system image, which includes the pre-configured ``salt-minion`` service.

Both components can be used independently of {productname}, or integrated with it. This
section describes how to use it with {productname}.

* For more information about using Yomi independently, see
https://github.com/openSUSE/yomi.
* For build assets, see https://build.opensuse.org/project/show/systemsmanagement:yomi.

To use Yomi for installing a client operating system, follow this process:

* Install the ``yomi-formula`` package.
* Prepare the Salt pillar for the new installation.
* Boot the new client using the PXE boot image for Yomi.


[NOTE]
====
To use Yomi with {productname}, ensure you have enough available memory. To boot from USB
or DVD image, you need at least 512{nbsp}MB. To boot from a PXE server, you need at least
2{nbsp}GB.
====



== Install the Yomi Formula
```

```
Before you begin, you need to install the Yomi formula, which is available as a package in
{productname}.

The ``yomi-formula`` package contains the Salt states and modules that describe the Yomi
state, and the formulas with forms to create the pillar. It also contains documentation
about the different sections of the pillar, and some examples about how to parameterize
installations based on {opensuse} or {sle}.

The formula package performs these actions:

* Adds a new configuration file called ``yomi-formula.conf`` in the
[path]``/etc/salt/master.d/`` directory. This configuration file defines the Python module
and Salt states required by Yomi.
* Installs the Yomi Salt states in the [path]``/usr/share/salt-formulas/states/``
directory.
* Provides some example configuration files in the [path]``/usr/share/yomi/`` directory.
* Installs the required forms and sub-forms in the [path]``/usr/share/salt-
formulas/metadata/`` directory.
* Provides some pillar examples in the [path]``/usr/share/yomi/pillar/`` directory.



.Procedure: Installing the Yomi Formula

. On the {productname} Server, at the command prompt, as root, install the ``yomi-
formula`` package:
+
```

zypper in yomi-formula

```
. Restart services:
+
```

systemctl restart salt-master.service

```
For more information about the Yomi formula, see xref:specialized-guides:salt/salt-
formula-yomi.adoc[].



== Install the PXE Image

To provision a new client, you need an operating system image to boot from. You can use
any image that contains a ``salt-minion`` service enabled, together with a minimal set of
tools that are required during the installation, for example ``parted`` or ``btrfstools``.

Yomi provides an already prepared image, based on openSUSE Tumbleweed, openSUSE Leap (for
{uyuni}), or SLE (for {susemgr}). For {productname}, the image is packaged as an RPM. This
is done in a similar way to how ``pxe-default-image`` is distributed.

The package installs a standard PXE OEM image generated by Kiwi, the initial kernel and
initrd in the [path]``/srv/pxe-yomi-image/`` directory, and the second stage kernel,
initrd and image in the [path]``/srv/pxe-yomi-image/image`` directory.



.Procedure: Installing the PXE Image


. On the {productname} Server, at the command prompt, as root, install the ``pxe-yomi-
```

```
image`` service:
+
```

zypper in pxe-yomi-image-sle15

```
When you have the package installed, you can register Yomi in {cobbler}.



== Register Yomi in {cobbler}

{productname} uses {cobbler} to manage the PXE boot service, so you will need to register
the image in {cobbler}.



.Procedure: Registering the Yomi Image in {cobbler}

. On the {productname} Server, at the command prompt, as root, create a directory for the
Yomi image:
+
```

mkdir /srv/tftpboot/pxe-yomi-image

```
. Define a distribution in {cobbler}, including the path to install the second stage
kernel and initrd, the location of the full image, and any further kernel options.
    Adjust this command to include the correct version of the product, and the TFTP server
address:
+
```

cobbler distro add \ --name=pxe-yomi-image \ --kernel=/srv/pxe-yomi-image/linux \ --initrd=/srv/pxe-yomi-image/initrd \ --boot-files='/srv/tftpboot/pxe-yomi-image/image.initrd=/srv/pxe-yomi-image/image/pxe-yomi-image-sle15.x86_64-1.0.0.initrd /srv/tftpboot/pxe-yomi-image/image.kernel=/srv/pxe-yomi-image/image/pxe-yomi-image-sle15.x86_64-1.0.0.kernel /srv/tftpboot/pxe-yomi-image/image.md5=/srv/pxe-yomi-image/image/pxe-yomi-image-sle15.x86_64-1.0.0.md5 /srv/tftpboot/pxe-yomi-image/image.config.bootoptions=/srv/pxe-yomi-image/image/pxe-yomi-image-sle15-x86_64-1.0.0.config.bootoptions /srv/tftpboot/pxe-yomi-image/image.xz=/srv/pxe-yomi-image/image/pxe-yomi-image-sle15.x86_64-1.0.0.xz' \ --kernel-options='rd.kiwi.install.pxe rd.kiwi.install.image=tftp://<server-address>/pxe-yomi-image/image.xz rd.kiwi.ramdisk ramdisk_size=2097152 net.ifnames=1'

```
+



By default, the ``salt-minion`` service in ``pxe-yomi-image`` is configured to find the
Salt master under the ``salt`` address. If the DNS server is not able to resolve this
address, you need to adjust the ``kernel-options`` parameter from the {cobbler} command
that register the distribution, and add a new kernel command line of
``ym.master=master_address``. This will override the default configuration for the ``salt-
minion``.


.Procedure: Registering the Yomi Profile in {cobbler}

. On the {productname} Server, at the command prompt, as root, define a profile in
{cobbler} based on the image.
```

```
+
```

cobbler profile add \ --name pxe-yomi-profile \ --distro=pxe-yomi-image

```
. OPTIONAL: Create a system in {cobbler}.
    If you know the MAC address for the new client to be provisioned, you can have it boot
directly from the Yomi image.
+
```

cobbler system add \ --name=yomi \ --mac=00:11:22:33:44:55 \ --profile=pxe-yomi-profile

```
. When the new node has been provisioned, remove the temporary {cobbler} system:
+
```

cobbler system remove --name=yomi

```
== Example Salt Pillar Preparation

The parameters of the new installation are defined with a Salt pillar. The pillar includes
parameters that the Yomi state requires during the installation, including the partitions,
file systems, repositories, packages installed, and services enabled.

The pillar is defined using the formulas with forms. In this example, we prepare the
pillar for a minimal openSUSE Tumbleweed installation. You can find examples for {sle} in
the example directory [path]``/usr/share/yomi/pillar/``.

To begin, boot the client that you want to provision using the Yomi PXE boot image, using
the {cobbler} procedures described earlier in this section.

When the ``salt-minion`` service is running on the new client, accept the key by
navigating to menu:Salt[Keys]. When the key is accepted, you can view and manage the
client by navigating to menu:Systems[Overview]. Navigate to the [guimenu]``Formulas`` tab,
and add all the Yomi Installer formulas to the client. When you have added all the
formulas, complete the forms and sub-forms. This section outlines each form and provides
example settings for a minimal installation. For a detailed explanation of every option,
see xref:specialized-guides:salt/salt-formula-yomi.adoc[].


Yomi::

The Yomi form contains some general configuration options. For example, the keyboard
language and layout, the locale information, and the option to perform a full reset of the
system after provisioning.

For this example, set the [parameter]``Reboot`` parameter to ``yes``.


Yomi Storage::

This sub-form provides information about the devices, partitioning, file system (including
the BtrFS subvolumes, for example), and LVM and RAID configuration.

For this example, we assume that the new client has a single device named ``/dev/sda``,
and that it belongs to a non-UEFI system. In this case, we have only three partitions: one
for the boot loader, one for swap and one for the system. We also expect to have an ext4
file system for the root directory.

Device 1:
```

```
* Device: /dev/sda
* Label: GPT
* Initial Gap: 1{nbsp}MB

Create three partitions:

* Partition 1:
** Partition Number: 1
** Partition Size: 1{nbsp}MB
** Partition Type: boot
* Partition 2:
** Partition Number: 2
** Partition Size: 1024{nbsp}MB
** Partition Type: swap
* Partition 3:
** Partition Number: 3
** Partition Size: rest
** Partition Type: linux

Create two file systems:

* Filesystem 1:
** Partition: /dev/sda2
** Filesystem: swap
* Filesystem 2:
** Partition: /dev/sda3
** Filesystem: ext4
** Mountpoint: /


Yomi Bootloader::

This sub-form provides details required for GRUB.

Set these parameters:

* Device: /dev/sda
* Theme: selected

The [parameter]``Kernel`` parameter can be used for the GRUB ``append`` section.


Yomi Software::

This form provides the different repositories and packages to install. You can also
register the product in this form, using SUSEConnect, and install the different modules
after registering.

For this example we are going to install a very minimal openSUSE Tumbleweed distribution,
using publicly available repositories. For production deployments, you will need to
provide a local repository.

Add a new repository:
* Repository Name: repo-oss
* Repository URL: http://download.opensuse.org/tumbleweed/repo/oss/

Add these packages:
* pattern:enhanced_base
* glibc-locale
* kernel-default

You can also add patterns and products, together with packages, by using the correct
prefix.
```

```
Yomi Services::

By default Yomi is installed with the ``salt-minion`` service, but you must enable it.

Add a new enabled service:

* Service 1:
** Service: salt-minion


Yomi Users::

This form sets out the system users. In this example, we have a single root user. To
provide a password, you must use the hashed version of the password, not the plain text.
This behavior is set to be changed in future versions of Yomi.

* User 1:
** Username: root
** Password Hash: $1$wYJUgpM5$RXMMeASDc035eXNbYWFl0



== Monitor the Installation

You can monitor the installation as it progresses, using the ``monitor`` tool from Yomi.
You can continue monitoring as the highstate is applied to the new client. To use the
tool, you will need to have enabled ``Events`` in the Yomi formula, and have the ``salt-
api`` service activated.

For more information about the ``salt-api`` service, and how to use the ``monitor`` tool,
see https://github.com/openSUSE/yomi.

:leveloffset: 2
:leveloffset: +2

[[config-modules]]
= Configuration Modules

Salt uses execution and state modules to define, apply, and orchestrate configuration of
your devices. {productname} provides a set of modules called {uyuni} configuration
modules, that you can use to configure both {susemgr} and {uyuni} Servers.

You can use the {uyuni} configuration modules directly or using SLS files. They are are
especially useful if you have multiple {productname} Servers, for example in Hub
installations, but they are also useful for smaller installations.

For more information about using Hub, see xref:specialized-guides:large-deployments/multi-
server.adoc[].

You can use {uyuni} configuration modules to configure:

* Organizations
* Users
* User permissions
* System groups
* Activation Keys


For more information about Salt execution modules, see
https://docs.saltproject.io/en/latest/topics/tutorials/modules.html.

For more information about Salt state modules, see
https://docs.saltproject.io/en/latest/topics/tutorials/starting_states.html.
```

```
== Install Configuration Modules

The {uyuni} configuration modules are available in the [package]``uyuni-config-modules``
package. On the {productname} Server, at the command prompt, as root, use this command:
```

zypper in uyuni-config-modules

```
This package also installs detailed API descriptions, indications on pillar settings, and
examples. When you have installed the package, navigate to
[package]``/usr/share/doc/packages/uyuni-config-modules/``.

:leveloffset: 2
:leveloffset: +2

[[salt.formulas]]
= Salt Formulas

Formulas are collections of Salt States that contain generic parameter fields. Formulas
allow for reliable reproduction of a specific configuration. Some formulas are supplied by
{suse}, or you can install formulas from RPM packages or an external git repository.

Formulas work best for large, non-trivial, configurations. For smaller tasks, use a state
rather than a formula. Formulas and states both act as a kind of configuration
documentation. When you have written and stored the configuration, they provide a snapshot
of your infrastructure.

Formula data can be managed using the XMLRPC API.

You can use the {productname} {webui} to apply {productname} formulas. The most commonly
used formulas are documented in this section.

Alternatively, you can use pre-written formulas as a starting point for your own custom
formulas. Pre-written formulas are available from https://github.com/saltstack-formulas.

For more information on custom formulas, see xref:specialized-guides:salt/salt-formulas-
custom.adoc[].

:leveloffset: 2
:leveloffset: +3

[[formulas-suma]]
= Formulas Provided by {productname}

Some formulas are installed by default with {productname}. Other official formulas can be
installed as RPM packages. When the formula is installed, you can activate them using the
{productname} {webui}.

For information about writing custom formulas, see xref:specialized-guides:salt/salt-
formulas-custom.adoc[].


== Install Formulas with Zypper

Formulas are provided in the {productname} pool software channel.


[IMPORTANT]
====
If a formula uses the same name as an existing Salt state, the two names will collide, and
could result in the formula being used instead of the state. Always check states and
formulas to avoid name clashes.
====
.Procedure: Installing Formulas with Zypper
. On the {productname} Server, at the command prompt, search for available formulas:
+
```

zypper se --type package formula

```
. Get more information about a formula:
+
```

zypper info <formula_name>

```
. On the {productname} Server, at the command prompt, as root, install the formula:
+
```

zypper in <formula_name>

```
== Activate Formulas from the {webui}

Formulas provided by {productname}, or formulas that you have installed, can be activated
using the {productname} {webui}.


.Procedure: Activate Formulas from the {webui}
. In the {productname} {webui}, navigate to menu:Systems[List], select the client you want
to activate the formula for.
. Navigate to the menu:Systems[Formulas] tab, and check the formula you want to activate.
. Click btn:[Save].
. Navigate to the new subtab for the formula, and configure the formula as required.
. Apply the highstate.

:leveloffset: 2
:leveloffset: +3

[[bind-formula]]
= Bind Formula

The Bind formula is used to configure the Domain Name System (DNS) on the branch server.
POS terminals will use the DNS on the branch server for name resolution of {saltboot}
specific hostnames.

When you are configuring the Bind formula for a branch server with a dedicated internal
network, check that you are using the same fully qualified domain name (FQDN) on both the
external and internal branch networks. If the FQDN does not match on both networks, the
branch server will not be recognized as a proxy server.

[NOTE]
====
The following procedure outlines a standard configuration with two zones. Adjust it to
suit your own environment.
====

Zone 1 is a regular domain zone. Its main purpose is to resolve {saltboot} hostnames such
as TFTP, FTP, or Salt. It can also resolve the terminal names if configured.

Zone 2 is the reverse zone of Zone 1. Its main purpose is to resolve IP addresses back to
hostnames. Zone 2 is primarily needed for the correct determination of the FQDNs of the
```

```
branch.

// REMARK: this procedure is probably too long.  Where to split?
// REMARK: are these list items (`*`) substeps?  Or what?  Confusing.
.Procedure: Configuring Bind with Two Zones

. Check the [systemitem]``Bind`` formula, click [btn]``Save``, and navigate to the
menu:Formulas[Bind] tab.
. In the [guimenu]``Config`` section, select [systemitem]``Include Forwarders``.
. In the [guimenu]``Configured Zones`` section, use these parameters for Zone 1:
* In the [guimenu]``Name`` field, enter the domain name of your branch network (for
example: [systemitem]``branch1.example.com``).
* In the [guimenu]``Type`` field, select [systemitem]``master``.
. Click [btn]``Add item`` to add a second zone, and set these parameters for Zone 2:
* In the [guimenu]``Name`` field, use the reverse zone for the configured IP range (for
example: [systemitem]``com.example.branch1``).
* In the [guimenu]``Type`` field, select [systemitem]``master``
. In the [guimenu]``Available Zones`` section, use these parameters for Zone 1:
* In the [guimenu]``Name`` field, enter the domain name of your branch network (for
example: [systemitem]``branch1.example.org``).
* In the [guimenu]``File`` field, type the name of your configuration file.
. In the [guimenu]``Start of Authority (SOA)`` section, use these parameters for Zone 1:
* In the [guimenu]``Nameserver (NS)`` field, use the FQDN of the branch server (for
example: [systemitem]``branchserver.branch1.example.org``).
* In the [guimenu]``Contact`` field, use the email address for the domain administrator.
* Keep all other fields as their default values.
. In the [guimenu]``Records`` section, in subsection [guimenu]``A``, use these parameters
to set up an A record for Zone 1:
* In the [guimenu]``Hostname`` field, use the hostname of the branch server (for example:
[systemitem]``branchserver``).
* In the [guimenu]``IP`` field, use the IP address of the branch server (for example,
[systemitem]``192.168.1.5``).
. In the [guimenu]``Records`` section, subsection [guimenu]``NS``, use these parameters to
set up an NS record for Zone 1:
* In the input box, use the hostname of the branch server (for example:
[systemitem]``branchserver``).
. In the [guimenu]``Records`` section, subsection [guimenu]``CNAME``, use these parameters
to set up CNAME records for Zone 1:
* In the [guimenu]``Key`` field, enter [systemitem]``tftp``, and in the [guimenu]``Value``
field, type the hostname of the branch server (for example: [systemitem]``branchserver``).
* Click [guimenu]``Add Item``. In the [guimenu]``Key`` field, enter [systemitem]``ftp``,
and in the [guimenu]``Value`` field, type the hostname of the branch server.
* Click [guimenu]``Add Item``. In the [guimenu]``Key`` field, enter [systemitem]``dns``,
and in the [guimenu]``Value`` field, type the hostname of the branch server.
* Click [guimenu]``Add Item``. In the [guimenu]``Key`` field, enter [systemitem]``dhcp``,
and in the [guimenu]``Value`` field, type the hostname of the branch server.
* Click [guimenu]``Add Item``. In the [guimenu]``Key`` field, enter [systemitem]``salt``,
and in the [guimenu]``Value`` field, type the FQDN of the branch server (for example:
[systemitem]``branchserver.branch1.example.org``).
. Set up Zone 2 using the same parameters as for Zone 1, but ensure you use the reverse
details:
* The same SOA section as Zone 1.
* Empty A and CNAME records.
*  Additionally, configure in Zone 2:
** `Generate Reverse` field by the network IP address set in branch server network formula
(for example, [systemitem]``192.168.1.5/24``).
** `For Zones` should specify the domain name of your branch network (for example,
[systemitem]``branch1.example.org``).
. Click btn:[Save Formula] to save your configuration.
. Apply the highstate.


[IMPORTANT]
====
Reverse name resolution on terminals might not work for networks that are inside one of
```

```
these IPv4 private address ranges:

* [systemitem]``10.0.0.0/8``
* [systemitem]``172.16.0.0/12``
* [systemitem]``192.168.0.0/16``

If you encounter this problem, go to the [guimenu]``Options`` section of the Bind formula,
and click btn:[Add item]:

* In the [guimenu]``Options`` field, enter [systemitem]``empty-zones-enable``.
* In the [guimenu]``Value`` field, select [systemitem]``No``.
====

:leveloffset: 2
:leveloffset: +3

[[branch-network-formula]]
= Branch Network Formula

The Branch Network formula is used to configure the networking services required by the
branch server, including DHCP, DNS, TFTP, PXE, and FTP.

== Set Up a Branch Server Networking

The branch server can be configured to use networking in many different ways. The most
common ways provide either a dedicated or shared LAN for terminals.

=== Set Up a Branch Server with a Dedicated LAN

In this configuration, the branch server requires at least two network interfaces: one
acts as a WAN to communicate with the {susemgr} server, and the other one acts as an
isolated LAN to communicate with terminals.

This configuration allows for the branch server to provide DHCP, DNS, TFTP, PXE, and FTP
services to terminals. These services can be configured with Salt formulas in the
{susemgr} {webui}.


.Procedure: Setting Up a Branch Server with a Dedicated LAN

. In the {susemgr} {webui}, open the details page for the branch server, and navigate to
the [guimenu]``Formulas`` tab.
. In the [guimenu]``Branch Network`` section, set these parameters:
* Keep [guimenu]``Dedicated NIC`` checked.
* In the [guimenu]``NIC`` field, enter the name of the network device that is connected to
the internal LAN.
* In the [guimenu]``IP`` field, enter the static IP address to be assigned to the branch
server on the internal LAN.
* In the [guimenu]``Netmask`` field, enter the network mask of the internal LAN.
. Check [guimenu]``Enable Route`` if you want the branch server to route traffic from
internal LAN to WAN.
* Check [guimenu]``Enable NAT`` if you want the branch server to convert addresses from
internal LAN to WAN.
* Select the [guimenu]``bind`` DNS forwarder mode.
* Check DNS forwarder fallback if you want to rely on an external DNS if the branch DNS
fails.
* Specify the working directory, and the directory owner and group.



=== Set up a Branch Server with a Shared Network

In this configuration, the branch server has only one network interface card, which is
used to connect to the {susemgr} server as well as the terminals.
```

```
This configuration allows for the branch server to provide DNS, TFTP, PXE, and FTP
services to terminals. These services can be configured with Salt formulas in the
{susemgr} {webui}. Optionally, the branch server can also provide DHCP services in this
configuration.

[NOTE]
====
If DHCP services are not provided by the branch server, ensure that your external DHCP
configuration is set correctly:

* The [systemitem]``next-server`` option must point to the branch server for PXE boot to
work.
* The [systemitem]``filename`` option must correctly identify the network boot program (by
default, this is [path]``/boot/pxelinux``).
* The [systemitem]``domain-name-servers`` option must point to the branch server for
correct host name resolution.
====


.Procedure: Setting Up a Branch Server with a Shared Network

. In the {susemgr} {webui}, open the details page for the branch server, and navigate to
the [guimenu]``Formulas`` tab.
. In the [guimenu]``Branch Network`` section, set these parameters:
* Keep [guimenu]``Dedicated NIC`` unchecked.
* Enable services on the branch server’ s firewall.
    Ensure you include DNS, TFTP, and FTP services.
* Select the [guimenu]``bind`` DNS forwarder mode.
* Check DNS forwarder fallback if you want to rely on an external DNS if the branch DNS
fails.
* Specify the working directory, and the directory owner and group.


== Set up Branch Server Terminal Naming

In this configuration it is required to fill at least [systemitem]``Branch
Identification``. This identifies Branch Server in Retail subsystem and is also used to
better organize terminals with their respective branch servers.

.Procedure: Setting up a Branch Server Identification

. In the {susemgr} {webui}, open the details page for the branch server, and navigate to
the [guimenu]``Formulas`` tab.
. In the [guimenu]``Terminal Naming`` section, enter the [systemitem]``Branch
Identification`` string.
. Click btn:[Save] to save your changes.
. Apply the highstate.

It is also possible to set various options about terminal naming, for more information
about terminal naming see xref:retail:retail-terminal-names.adoc[].

:leveloffset: 2
:leveloffset: +3

[[dhcpd-formula]]
= DHCPd Formula

The DHCPd formula is used to configure the DHCP service on the branch server.


.Procedure: Configuring DHCP

. In the {susemgr} {webui}, open the details page for the branch server, and navigate to
the Formulas tab.
. Check the [guimenu]``Dhcpd`` formula, and click btn:[Save].
. Navigate to the menu:Formulas[Dhcpd] tab, and set these parameters:
* In the [guimenu]``Domain Name`` field, enter the domain name for the branch server (for
example: [systemitem]``branch1.example.com``).
* In the [guimenu]``Domain Name Server`` field, enter either the IP address or resolvable
FQDN of the branch DNS server (for example: [systemitem]``192.168.1.5``).
* In the [guimenu]``Listen Interfaces`` field, enter the name of the network interface
used to connect to the local branch network (for example: [systemitem]``eth1``).
. Navigate to the [guimenu]``Network Configuration (subnet)`` section, and use these
parameters for Network1:
* In the [guimenu]``Network IP`` field, enter the IP address of the branch server network
(for example: [systemitem]``192.168.1.0``).
* In the [guimenu]``Netmask`` field, enter the network mask of the branch server network
(for example: [systemitem]``255.255.255.0``).
* In the [guimenu]``Domain Name`` field, enter the domain name for the branch server
network (for example: [guimenu]``branch1.example.com``).
. In the [guimenu]``Dynamic IP Range`` section, use these parameters to configure the IP
range to be served by the DHCP service:
* In the first input box, set the lower bound of the IP range (for example:
[systemitem]``192.168.1.51``).
* In the second input box, set the upper bound of the IP range (for example:
[systemitem]``192.168.1.151``).
. In the [guimenu]``Broadcast Address`` field, enter the broadcast IP address for the
branch network (for example: [systemitem]``192.168.1.255``).
. In the [guimenu]``Routers`` field, enter the IP address to be used by routers in the
branch server network (for example: [systemitem]``192.168.1.5``).
. In the [guimenu]``Next Server`` field, enter the hostname or IP address of the branch
server (for example: [systemitem]``192.168.1.5``).
. In the [guimenu]``Filename`` field, if you are booting a client using PXE, type the path
to the PXE bootloader.
    There is usually no need to change the default value of
[systemitem]``/boot/pxelinux.0``.
. In the [guimenu]``Filename Efi`` field, if you are booting a UEFI client using PXE, type
the path to the PXE bootloader.
    There is usually no need to change the default value of
[systemitem]``/boot/shim.efi``.
. In the [guimenu]``Filename Http`` field, if you are booting a UEFI client using HTTP,
type **+http://branchserver/saltboot/boot/shim.efi+**.
. Click btn:[Save Formula] to save your configuration.
. Apply the highstate.

:leveloffset: 2
:leveloffset: +3

[[image-sync-formula]]
= Image Synchronization Formula

The Image Synchronization formula is used to configure when OS images are synchronized to
the branch server, and to specify which images to synchronize.

If this formula is not enabled, synchronization must be started manually, and all images
will be synchronized.


.Procedure: Configuring Image Synchronization

. In the {susemgr} {webui}, open the details page for the branch server, and navigate to
the Formulas tab.
. Check the [guimenu]``Image Synchronize`` formula, and click btn:[Save].
. Navigate to the menu:Formulas[Image Synchronize] tab, and set these parameters:
* Check the [guimenu]``Include Image Synchronization in Highstate`` field to have image
synchronization occur every time highstate is applied.
    This ensures that you do not have to perform image synchronization manually, however
it requires a high bandwidth environment.
* In the [guimenu]``Synchronize only the listed images`` field, click btn:[Add item] to
add the images you want to have synchronized automatically.
```

```
    Alternatively, you can leave this list blank to have all images synchronized.
. Click btn:[Save Formula] to save your configuration.
. Apply the highstate.


[NOTE]
====
The Image Synchronization state does not delete cached images. If you are running out of
disk space, check the size of the Salt client cache directory, and delete it if required.
By default, the directory is located at ``/var/cache/salt/minion``.
====

:leveloffset: 2
:leveloffset: +3

// FIXME: where appropriate, use the "Confirm with" expression.

[[liberate-formula]]
= Liberate Formula

The liberate formula migrates systems from EL clients such as {centos} 7 or {rhel} 9 to
{sll}. With this formula the conversion will take place during the client onboarding on
{productname}.



== Install the formula

Install the formula with:
```

zypper in liberate-formula

```
== Configure {productname}



To provide all the software channels for {sll} on the {productname} Server, proceed as
follows.



.Procedure: Providing SUSE Customer Center credentials

. Sign in to {scc} at https://scc.suse.com.

. Navigate to [guimenu]``My Organization``, and select your organization.

. Navigate to menu:Users[Organization Credentials] and take note of your organization
username and password.

. In the {productname} {webui}, navigate to menu:Admin[Setup Wizard > Organization
Credentials] to add the credentials to your {productname} Server.

. Click btn:[Add new credential], and enter the {scc} username and password noted in a
previous step.


.Procedure: Synchronizing the SLL/SLES-ES channels:

. In the {productname} {webui}, navigate to menu:Admin[Setup Wizard > Products]
. Select the {sll} Channels that you will use:
+
* EL7 LTSS: `SUSE Linux Enterprise Server with Expanded Support LTSS 7 x86_64`
* EL7: `SUSE Linux Enterprise Server with Expanded Support 7 x86_64`
* EL8: `RHEL or SLES ES or CentOS 8 Base`
* EL9: `RHEL and Liberty 9 Base`

. Click the top right button btn:[Add products].


Initial sychronization can take considerable time. You can check progress by accessing the
server machine via SSH and monitoring the logs using:
```

tail -f /var/log/rhn/reposync/*

```
.Procedure: Creating one Activation Key per {sll} parent channel

. Note: Activation Keys are the way to register systems and automatically assign them to
the required software and configuration channels corresponding to them.
. In the {productname} {webui}, navigate to menu:Systems[Activation Keys], and click the
top right button btn:[Create key].
. In the [guimenu]``Activation Key`` dialog, set the fields:
+
Description::
Enter some text describing the activation key.
Key::
Enter the identifier of the key. For example [literal``sll9-default`` for EL 9 systems.
Note: Keys will have a numeric prefix depending on the organization, so that they are
unique.
Usage::
Leave blank.
Base Channel::
Select one base channel:
+

* EL7 LTSS: `RES-7-LTSS-Updates for x86_64`
* EL7: `RHEL x86_64 Server 7`
* EL8: `RHEL8-Pool for x86_64`
* EL9: `EL9-Pool for x86_64`

Child Channel::
Include all child channels.

Add-On system type::
Leave all blank.
Contact Method::
Default
Universal Default::
Leave unchecked.

. Click btn:[Create Activation Key].



== Add Liberate formula and assign it to activation keys

When installed, the formula can be assigned to an Activation Key by creating a System
Group:

.Procedure: Assigning system group and assigning liberate formula

. In the {productname} {webui}, navigate to menu:Systems[System Groups], and click the
btn:[Create Group] button in top right corner.

. In the dialog, fill in the following data:
```

```
+

Name::
liberate
Description::
Systems to be converted to SUSE Liberty Linux

. From the `liberate` System Group page, navigate go to the [guimenu]``Formulas`` tab.

. Select the [guimenu]``Liberate`` formula, and click btn:[Save]. A new tab called
`Liberate` will appear.

. On the [guimenu]``Liberate`` tab, you see the [option]``Reinstall all packages after
conversion`` option. Keep it checked if you want to reinstall all the packages during the
migration. This way you ensure all the packages will have {suse} signatures and no
previous package will be kept. If you do not want to change the state of your system
during the migration, uncheck this option and click the btn:[Save Formula] button. In this
case, you can re-install the packages later.

Now a system group exists that has assigned the [guimenu]``Liberate`` formula. This
formula will be applied only once to migrate the system to {sll}, even if you run it
multiple times. With the next procedure, assign the system group to the Activation Key.


.Procedure: Assigning the system group to the Activation Key

. In the {productname} {webui}, navigate to menu:Systems[Activation Keys].

. Select the Activation Key, for example [literal]``sll9-default`` for the EL 9 systems.

. From the Activation Key page navigate to the menu:Groups[Join] tab, select the
[literal]``liberate`` group, and click the btn:[Join Selected Groups] button. The group
will be assigned to the Activation Key

.Procedure: Applying migrate directly during registration

. From the Activation Key page, navigate to the [guimenu]``Details`` tab.

. Navigate to the [guimenu]``Configuration File Deployment`` section, and checkb the
[option]``Deploy configuration files to systems on registration``option.

. Click btn:[Update Activation Key].

When you register a system with this key it will perform the migration automatically.



== Register a new system and proceed to the migration

There are two ways to onboard (or register) a new client with the Activation Key:

With the {webui} and selecting the activation key::
This is intended for a one-off registration or for testing purposes.

With a bootstrap script with an assigned activation key::
This is intended to be used for mass registration.



=== Register with the {webui}

Technically, this will start an SSH connection to the client and run the bootstrap script
to register it.

.Procedure: Registering with the {webui} and selecting the activation key
```

```
. In the {productname} {webui}, navigate to menu:Systems[Bootstraping].

. In the `Bootstrap Minions` dialog, fill the entries:
+

Host::
Hostname of the client to register
SSH Port::
Leave blank to use default, which is [literal]``22``
User::
Enter user or leave blank for {rootuser}
Authentication Method::
Select if you want to use [guimenu]``password`` or provide a [guimenu]``SSH Private Key``
+
  * [guimenu]``Password``: password to access the system
  * [guimenu]``SSH Private Key``: file with the private key
  * [guimenu]``SSH Private Key Passphrase``: In case a private key was provided that
requires a passphrase to unlock, provide it here
Activation Key::
Select from the menu the activation key to be used, for example [literal]``sll9-default``
Reactivation Key::
Leave blank, it will not be used here
Proxy::
Leave as [literal]``None`` if you are not using a proxy

. Click the btn:[Bootstrap] button to start the registration.

A notification will show on top of the page stating that the client is being registered.



=== Register with a bootstrap script


.Procedure: Creating bootstrap script

. In the {productname} {webui}, navigate to menu:Admin[Manager Configuration > Bootstrap
Script].

. Fill the fields of the bootstrap script configuration dialog:
+
Uyuni/SUSE Manager server hostname::
This should be set to the hostname that the client will use to reach the server, as well
as the hostname Note: a Certificate will be used associated to this name for the client
systems, as it was configured in the initial setup. If it's changed, a new certificate
shall be created
SSL cert location::
Path, in the server, to the filename provided as a certificate to register it. Keep it as
it is.
Bootstrap using Salt::
Select this checkbox to apply salt states, like the one we added via configuration
channel. It is required to perform the migration.
Enable Client GPG checking::
Select this checkbox to ensure all packages installed come from the proper sources, in
this case, {sll} signed packages.
Enable Remote Configuration::
Leave unchecked.
Enable Remote Commands::
Leave unchecked.
Client HTTP Proxy::
Leave blank. This is in case the client requires a proxy to access the server.
Client HTTP Proxy username::
Leave blank.
Client HTTP Proxy password::
```

```
Leave blank.

. Click the btn:[Update] button to refresh the [path]``bootstrap.sh`` script.


The bootstrap script generated is reachable via HTTP. For example, for a server named
[literal]``example.org`` it will be at **+https://example.org/pub/bootstrap/+**. Accessing
the server via SSH, the bootstrap script is available in
[path]``/srv/www/htdocs/pub/bootstrap/``.


.Procedure: Modifying and running the bootstrap script

. Make a copy of the bootstrap script generated on the server in
[path]``/srv/www/htdocs/pub/bootstrap/``. For example:
+
```

cd /srv/www/htdocs/pub/bootstrap/ cp bootstrap.sh bootstrap-sll9.sh

```
. Edit [path]``bootstrap-sll9.sh``, and add the activation key created earlier. For
example, [literal]``sll9-default``:
+
```

ACTIVATION_KEYS=sll9-default

```
. Run the modified [path]``bootstrap-sll9.sh`` script. On the client, as root, run:
+
```

curl -Sks https://example.org/pub/bootstrap/bootstrap-sll9.sh | /bin/bash

```
+

  For more information about running bootstrap scripts, see xref:client-
configuration:registration-bootstrap.adoc#registering.clients.bootstrap.register[].
+

// Configuration channel and software channels will be assigned automatically by the
Activation Key

. Apply highstate to migrate the client to {sll}.

// The high state apply will both apply the configuration channel and migrate the machine
to Liberty Linux



== For already registered clients

Software channels, system group membership, and formulas can be assigned to any already
registered client. This method makes use of the bootstrap script created above for
onboarding new systems.

. In the {productname} {webui}, open the System Details page of any registered client you
want to migrate to {sll}.

. Click the [guimenu]``Reactivation`` tab. If there is already a key listed, you can use
it. If not, click btn:[Generate New Key], and copy the entire key. The key will start with
[literal]``re-``.
```

```
. SSH into this client and set the environment variable to be the key that you copied:
```

export REACTIVATION_KEY=re-xxxxxxxxxxxxxx

```
. Run the bootstrap script you created above, and the system will re-register using the
same profile as before, but with the newly assigned {sll} context.

////
## Version testing status

| OS version  | Status  |
| ----------- | ------- |
| Rhel 9      | Working |
| Rocky 9     | Working |
| Alma 9      | Working |
| Oracle 9    | Working |
| Rhel 8      | Working |
| Rocky 8     | Working |
| Alma 8      | Working |
| Oracle 8    | Working |
| Rhel 7      | Not Tested |
| CentOS 7    | Working |
| Oracle 7    | Working |
////

:leveloffset: 2
:leveloffset: +3

[[monitoring-formula]]
= Monitoring Formula

The monitoring services in {productname} are configured using formulas with forms. The
package is installed by default, and contains these formulas:

* Grafana
* Prometheus
* Prometheus Exporters


For more information about using monitoring, see xref:administration:monitoring.adoc[].



== Grafana




.Procedure: Configuring the Grafana Formula
. Navigate to the menu:Formulas[Grafana] tab, and set these parameters in the
[guimenu]``Grafana`` section:
* Check the [guimenu]``Enabled`` box to enable Grafana visualizations.
+
[NOTE]
====
[guimenu]``Initial admin password`` is used on the first run only, and Grafana UI prompts
to change it. This field cannot be used to change the Grafana password. For more
information on how to change the password, see xref:administration:monitoring.adoc[].
====
. For each Prometheus data source you want to use, in the menu:Datasources[Prometheus]
section, click btn:[+], and set these parameters:
* In the [guimenu]``Datasource name`` field, type a name to identify the data source.
* In the [guimenu]``Prometheus URL`` field, type the used protocol, the location of the
```

```
Prometheus server, and append port ``9090``.
    For example, ``http://example.com:9090``. In case TLS encryption is enabled in
Prometheus formula make sure to use `https` protocol and FQDN.
* In the fields [guimenu]``Prometheus server username`` and [guimenu]``Prometheus server
password``,
    enter basic authentication credentials for Prometheus server matching the ones in
Prometheus formula.
. In the [guimenu]``Dashboards`` section, check the dashboards you want to use:
* [guimenu]``Uyuni server dashboard``
* [guimenu]``Uyuni clients dashboard``
* [guimenu]``PostgreSQL dashboard``
* [guimenu]``Apache HTTPD dashboard``
* [guimenu]``Kubernetes cluster dashboard``
* [guimenu]``Kubernetes etcd dashboard``
* [guimenu]``Kubernetes namespaces dashboard``
. Click btn:[Save Formula] to save your configuration.



== Prometheus

.Procedure: Configuring the Prometheus Formula
. Navigate to the menu:Formulas[Prometheus] tab, and set these parameters in the
[guimenu]``Prometheus`` section:
* Check the [guimenu]``Enabled`` box to enable Prometheus monitoring.
* In the [guimenu]``Scrape interval`` field, type the frequency of data scraping, in
seconds.
    For example, ``15`` will scrape data every fifteen seconds.
* In the [guimenu]``Evaluation interval`` field, type the frequency of rules evaluation,
in seconds.
    For example, ``15`` will evaluate alerting and aggregation rules every fifteen
seconds.
. In the [guimenu]``TLS`` section, set these parameters:
* Check the [guimenu]``Enabled`` box to enable the secure configuration on Prometheus
server.
* In the [guimenu]``Server Certificate`` field, type the path to the TLS server
certificate.
* In the [guimenu]``Server Key`` field, type the path to the TLS server key.
* In the [guimenu]``User`` field, type the user name for Prometheus server.
* In the [guimenu]``Password Hash`` field, type the password for Prometheus server hashed
with bcrypt.
. In the [guimenu]``Uyuni Server`` section, set these parameters:
* Check the [guimenu]``Enabled`` box to enable monitoring on this server.
* Check the [guimenu]``Autodiscover clients`` box to enable Prometheus to automatically
find and monitor new clients when they are added to the server.
* In the [guimenu]``Username`` field, type the user name of the Prometheus account on the
server.
* In the [guimenu]``Password`` field, type the password of the Prometheus account on the
server.
* In the [guimenu]``Targets TLS`` section, set these parameters:
** Check the [guimenu]``Enabled`` box to enable the secure configuration for auto-
discovered targets.
** In the [guimenu]``CA Certificate`` field, type the path to the Certificate Authority
certificate.
** In the [guimenu]``Client Certificate`` field, type the path to the TLS client
certificate for authentication.
** In the [guimenu]``Client Key`` field, type the path to the TLS client key for
authentication.
. In the [guimenu]``Alerting`` section, set these parameters:
* Check the [guimenu]``Enable local Alertmanager service`` box to enable the alert manager
service.
* Check the [guimenu]``Use local Alertmanager`` box to use the local alert manager
service.
. For each alert manager you want to use, in the menu:Alerting[Alertmanagers] section,
click btn:[+], and set these parameters:
```

```
* In the [guimenu]``IP Address:Port`` field, type the location of the alert manager
target, including the port number.
//For example, ``FIXME``.
. To use a rule file, in the menu:Alerting[Rule Files] section, click btn:[+], and set
these parameters:
* In the [guimenu]``Rule Files`` field, type the location of the rule file you want to
use.
//For example, ``FIXME``.
. To add a custom scrape configuration, in the [guimenu]``User defined scrape
configurations`` section, click btn:[+], and set these parameters:
* In the [guimenu]``Job name`` field, type a unique job name for your configuration.
* In the [guimenu]``Files`` field, type the location pattern of file service discovery
files you want to use. For more information, see the upstream documentation
https://prometheus.io/docs/prometheus/latest/configuration/configuration/#file_sd_config.
. Click btn:[Save Formula] to save your configuration.


[NOTE]
====
The formula does not generate and deploy the TLS certificates and keys. Ensure the files
are present on the Salt client and readable for the user ``prometheus`` before applying
the highstate. For more information about generating client and server certificates, see
xref:administration:monitoring.adoc[].
====



== Prometheus Exporters

.Procedure: Configuring the Prometheus Exporters Formula
. Navigate to the menu:Formulas[Prometheus Exporters] tab, and set these parameters in the
[guimenu]``Node Exporter`` section:
* Check the [guimenu]``Enabled`` box to enable the node exporter.
* In the [guimenu]``Arguments`` field, type any customized arguments for this exporter.
    For example, ``--web.listen-address=":9100"``.
. In the [guimenu]``Apache Exporter`` section:
* Check the [guimenu]``Enabled`` box to enable the Apache exporter.
* In the [guimenu]``Arguments`` field, type any customized arguments for this exporter.
    For example, ``--telemetry.address=":9117"``.
. In the [guimenu]``Postgres Exporter`` section:
* Check the [guimenu]``Enabled`` box to enable the PostreSQL exporter.
* In the [guimenu]``Data source Name`` field, type the name of the data source to use.
* In the [guimenu]``Arguments`` field, type any customized arguments for this exporter.
    For example, ``--web.listen-address=":9187"``.
. In the [guimenu]``TLS`` section:
* Check the [guimenu]``Enabled`` box to enable the secure configuration.
* In the [guimenu]``CA Certificate`` field, type the path to the Certificate Authority
certificate.
* In the [guimenu]``Server Certificate`` field, type the path to the TLS server
certificate.
* In the [guimenu]``Server Key`` field, type the path to the TLS server key.
. Click btn:[Save Formula] to save your configuration.



=== File-based service discovery

It is possible to define monitored targets using file-based service discovery provided in
the Prometheus formula. This is a basic example demonstrating the usage:
```

[{ "targets": ["<client1>:9100", "<client2>:9100"], "labels": { "role": "<suma-client>", "job": "<suma-refclient>" } }, { "targets": ["<server>:80"], "labels": { "role": "<suma-server>", "job": "<suma-refhost>", "metrics_path": "/rhn/metrics" } }]

```
For more information, see https://prometheus.io/docs/guides/file-sd/.



=== TLS certificates and keys

The formula does not generate and deploy the TLS certificates and keys. Ensure the files
are present on the Salt client and readable for the user ``prometheus`` before applying
the highstate. For more information about generating client and server certificates, see
xref:administration:monitoring.adoc[].



== Activate Forms

When you have completed and saved all the forms, apply the highstate.

For more information about using monitoring, see xref:administration:monitoring.adoc[].

:leveloffset: 2
:leveloffset: +3

[[pxe-formula]]
= PXE Formula

The PXE formula is used to configure PXE booting on the branch server.


.Procedure: Configuring PXE Booting
. In the {susemgr} {webui}, open the details page for the branch server, and navigate to
the [guimenu]``Formulas`` tab.
. Select the [systemitem]``Pxe`` formula, and click [btn]``Save``.
. Navigate to the menu:Formulas[Pxe] tab, and set these parameters:
* In the [guimenu]``Kernel Filename`` field, keep the default value.
* In the [guimenu]``Initrd Filename`` field, keep the default value.
* If the terminals connecting to this branch server are running {arm64} architecture,
check the [guimenu]``Enable ARM64 UEFI boot`` box. Leave unchecked for {x86_64}.
* In the [guimenu]``Kernel Filename for ARM64`` field, keep the default value.
* In the [guimenu]``Initrd Filename for ARM64`` field, keep the default value.
* In the [guimenu]``Kernel Command Line Parameters`` field, keep the default value. For
more information about possible values, see <<retail.sect.formulas.pxe.kernelparams>>.
* In the [guimenu]``PXE root directory`` field, enter the path to the {saltboot} directory
(for example, [systemitem]``/srv/saltboot``).
. Click [btn]``Save Formula`` to save your configuration.
. Apply the highstate.



[[retail.sect.formulas.pxe.kernelparams]]
== {saltboot} Kernel Command Line Parameters

{saltboot} supports common kernel parameters and {saltboot}-specific kernel parameters.
All the parameters can be entered in the [guimenu]``Kernel Command Line Parameters`` field
of the PXE formula.

[systemitem]``kiwidebug=1``::
Starts a shell on tty2 during boot and enables debug logging in Salt.
+
[WARNING]
====
Do not use this parameter in a production environment as it creates a major security hole.
This parameter should be used only in a development environment for debug purposes.
====
[systemitem]``MASTER``::
Overrides auto-detection of the Salt master. For example:
+
```

MASTER=myproxy.domain.com

```
[systemitem]``SALT_TIMEOUT``::
Overrides the local boot fallback timeout if the Salt master does not apply the {saltboot}
state within this timeout (default: 60 seconds). For example:
+
```

SALT_TIMEOUT=300

```
[systemitem]``DISABLE_HOSTNAME_ID``::
If the terminal has a hostname assigned by DHCP, it is by default used as a minion ID.
Setting this option to `1` disables this mechanism, and SMBios information will be used as
a minion ID.

[systemitem]``DISABLE_UNIQUE_SUFFIX``::
Setting this option to `1` disables adding random generated suffix to terminal minion ID.
+
If you set this parameter make sure your terminal has either a unique hostname provided by
DHCP and DNS, or the terminal hardware comes with a unique serial number stored in its
SMBios memory. Otherwise there is a risk of terminal minion ID duplicity, and
bootstrapping the minion will fail.

The following parameters (`MINION_ID_PREFIX`, `salt_device`, `root`) are usually
autoconfigured and should be used only in specific conditions such as debugging or
development:

[systemitem]``MINION_ID_PREFIX``::
Branch ID set in the Branch Network formula form.

[systemitem]``salt_device``::
Device that contains the Salt configuration.

[systemitem]``root``::
Device that contains the already deployed root file system. Used for falling back to local
boot.

:leveloffset: 2
:leveloffset: +3

[[saltboot-formula]]
= {saltboot} Formula

The {saltboot} formula is used to configure disk images and partitioning for the selected
hardware type.

[IMPORTANT]
====
The {saltboot} formula is meant to be used as a group formula. Enable and configure
{saltboot} formula for hardware type groups.
====

[IMPORTANT]
====
To apply changes to a terminal, terminal needs to be restarted. Applying highstate does
not have any effect on running terminals.
====
.Procedure: Configuring the Hardware Type Group with {saltboot}

. Open the details page for your new hardware type group, and navigate to the
[guimenu]``Formulas`` tab.
. Select the {saltboot} formula and click btn:[Save].
. Navigate to the menu:Formulas[Saltboot] tab.
. In the [guimenu]``Disk 1`` section, set these parameters:
* In the [guimenu]``Disk symbolic ID`` field, enter a custom name for the disk (for
example, [systemitem]``disk1``).
* In the [guimenu]``Device type`` field, select [systemitem]``DISK``.
* In the [guimenu]``Disk device`` field, select the device that corresponds to the device
name on the target machine or asterisk [systemitem]``*``, see xref:saltboot-formula-disk-
selection[].
* In the [guimenu]``RAID level`` field, leave it empty.
* In the [guimenu]``Disk Label`` field, select [systemitem]``gpt``.
. In the [guimenu]``Partition`` section, set these parameters for [guimenu]``Partition
1``:
* In the [guimenu]``Partition symbolic ID`` field, enter a custom name for the partition
(for example, [systemitem]``p1``).
* In the [guimenu]``Partition size`` use value 500.
* In the [guimenu]``Device mount point`` use [path]``/boot/efi``.
* In the [guimenu]``Filesystem format`` use [systemitem]``vfat``.
* In the [guimenu]``OS Image to deploy`` field, leave it empty.
* In the [guimenu]``Partition encryption password`` field, leave it empty.
* In the [guimenu]``Partition flags`` use [systemitem]``boot``.
. In the [guimenu]``Partition`` section, set these parameters for [guimenu]``Partition
2``:
* In the [guimenu]``Partition symbolic ID`` field, enter a custom name for the partition
(for example, [systemitem]``p2``).
* In the [guimenu]``Partition size`` field, specify a size for the partition in Mebibytes
(MiB).
* In the [guimenu]``Device mount point`` field, select a location to mount the partition
(for example, [path]``/data``).
* In the [guimenu]``Filesystem format`` field, select your preferred format (for example,
[systemitem]``xfs``).
* In the [guimenu]``OS Image to deploy`` field, leave it empty.
* In the [guimenu]``Partition encryption password`` field, enter a password if you want to
encrypt the partition.
* In the [guimenu]``Partition flags`` field, leave it empty.
. In the [guimenu]``Partition`` section, set these parameters for [guimenu]``Partition
3``:
* In the [guimenu]``Partition symbolic ID`` field, enter a custom name for the partition
(for example, [systemitem]``p3``).
* In the [guimenu]``Partition size`` field, specify a size for the partition in Mebibytes
(MiB).
* In the [guimenu]``Device mount point`` field, leave it empty.
* In the [guimenu]``Filesystem format`` field, select [systemitem]``swap``.
* In the [guimenu]``OS Image to deploy`` field, leave it empty.
* In the [guimenu]``Partition encryption password`` field, enter a password if you want to
encrypt the partition.
* In the [guimenu]``Partition flags`` field, select [systemitem]``swap``.
. In the [guimenu]``Partition`` section, set these parameters for [guimenu]``Partition
4``:
* In the [guimenu]``Partition symbolic ID`` field, enter a custom name for the partition
(for example, [systemitem]``p4``).
* In the [guimenu]``Partition size`` field, leave it empty.
    This will ensure the partition uses up all remaining space.
* In the [guimenu]``Device mount point`` field, select [systemitem]``/``.
* In the [guimenu]``Filesystem format`` field, leave it empty.
* In the [guimenu]``OS Image to deploy`` field, enter the name of the image to deploy.
* In the [guimenu]``Image version`` field, leave it empty.
    This will ensure you use the latest available version.
* In the [guimenu]``Partition encryption password`` field, enter a password if you want to
encrypt the partition.
* In the [guimenu]``Partition flags`` field, leave it empty.
```

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



[[saltboot-formula-partition-types]]
== Special Partition Types

The {saltboot} formula helps you with setting up special partition types.

[IMPORTANT]
====
For terminal to be able to boot locally, either [systemitem]``BIOS grub`` or
[systemitem]``EFI`` partition must be configured.
====

=== BIOS grub Partition

A BIOS grub partition is needed for local booting from a `GPT` disk on non-EFI machines.
For more information, see https://en.wikipedia.org/wiki/BIOS_boot_partition.

In the formula, enter the following options:
```

Partition Symbolic ID: p1 Partition Size (MiB): 50 Partition Flags: bios_grub

```
Leave the other fields empty.



=== EFI Partition

An EFI partition is needed for local booting on EFI machines, [guimenu]``Partition Table
Type`` must be `GPT`. For more information, see
https://en.wikipedia.org/wiki/EFI_system_partition.

In the formula, enter the following options:
```

Partition Symbolic ID: p1 Partition Size (MiB): 500 Device Mount Point: /boot/efi Filesystem Format: vfat Partition Flags: boot

```
Leave the other fields empty.

[[saltboot-formula-disk-selection]]
== Disk Selection in {saltboot} Formula

When there is only one disk present on target hardware (including USB drives), use an
asterisk [systemitem]``*`` to automatically select the disk device.

When there are multiple disks, use an asterisk [systemitem]``*`` in the device path. In
this example, SATA disks are differentiated from USB disks:
```

/dev/disk/by-path/**-ata-1 /dev/disk/by-path/*usb**

```
If the entered value does not contain [systemitem]``/``, the entered value is
automatically prepended by [path]``/dev/disk/by-path/``. For example,
[systemitem]``\*usb*`` is the same as [path]``/dev/disk/by-path/\*usb*``.

If you prefer to select specific devices, you can this format in the disk device field:
```

```
* symbolic names (for example: [path]``/dev/sda``)
* by-path (for example: [path]``/dev/disk/by-path/..``)
* by-id (for example: [path]``/dev/disk/by-id/...``)

To see a list of available devices from the command prompt, press kbd:[Esc] while waiting
for key approval.



== Troubleshooting the {saltboot} Formula

``msdos`` Disklabel Limitations::

On the ``msdos`` disk label, you can create a maximum of four primary partitions. Extended
partitions are not supported. If you need more than four partitions, use the ``GPT`` disk
label instead.

For more information on troubleshooting problems with the {saltboot} formula, see
xref:administration:troubleshooting/tshoot-saltboot.adoc[].

:leveloffset: 2
:leveloffset: +3

[[tftpd-formula]]
= TFTPd Formula

The TFTPd formula is used to configure the TFTP service on the {productname} {smr} branch
server.



.Procedure: Configuring TFTP

. In the {susemgr} {webui}, open the details page for the branch server, and navigate to
the [guimenu]``Formulas`` tab.
. Select the [systemitem]``Tftpd`` formula, and click btn:[Save].
. Navigate to the menu:Formulas[Tftpd] tab, and set these parameters:
* In the [guimenu]``Internal Network Address`` field, enter the IP address of the branch
server (for example: [systemitem]``192.168.1.5``).
* In the [guimenu]``TFTP Base Directory`` field, enter the path to the {saltboot}
directory (for example, [systemitem]``/srv/saltboot``).
* In the [guimenu]``Run TFTP Under User`` field, enter [systemitem]``saltboot``.
. Click btn:[Save Formula] to save your configuration.
. Apply the highstate.

:leveloffset: 2
:leveloffset: +3

[[virt-guest-formula]]
= Virtualization Guest Formula

The Virtualization Guest formula is used to configure settings for virtual machine.

Open the [guimenu]``Tuning`` pull-down checklist and select from the following virtual
machine performance tuning settings:

* [guimenu]``Disable IRQ balancing``.
* [guimenu]``Disable Kernel Samepage Merging (KSM)``. Reduces performance overhead by not
sharing memory across virtual machines.
* [guimenu]``Optimizations for KVM passed through host CPU``. Requires the KVM hint-
dedicated option to be set on the VM definition.

When configured, click btn:[Save Formula] to save your configuration, and apply the
highstate.
```

```
:leveloffset: 2
:leveloffset: +3

[[virt-host-formula]]
= Virtualization Host Formula

The Virtualization Host formula is used to configure settings for a virtualization host.

Hypervisor::
Select KVM or Xen as the hypervisor.

Create default storage pool::

Create default virtual network::
+

* Default pool: Open the pull-down list and enter the directory name of the default
storage pool.
* Default net: Open the pull-down list and configure the default virtual network by
setting the [guimenu]``Mode`` ([literal]``NAT`` or [literal]``Bridge``) and the
[guimenu]``Bridge name``.

Tuning::
Enable IOMMU support (x86_64).

When configured, click btn:[Save Formula] to save your configuration, and apply the
highstate.

:leveloffset: 2
:leveloffset: +3

[[yomi-formula]]
= Yomi Formula

The Yomi (yet one more installer) installer for {suse} and openSUSE operating systems is
configured using formulas with forms.

The ``yomi-formula`` package provides these formulas:

* Yomi
* Yomi Storage
* Yomi Bootloader
* Yomi Software
* Yomi Services
* Yomi Users



.Procedure: Install the Yomi Formulas with Forms
. On the {productname} Server, at the command prompt, as root, install the ``yomi-
formula`` package:
+
```

zypper in yomi-formula

```
. Restart services:
+
```

systemctl restart salt-master.service

```
When the formula package is installed, you need to install the PXE Yomi image on the
```

```
client, boot the client you want to provision, and enable the Yomi formulas on the client.
For more information on preparing Yomi clients for provisioning, see xref:specialized-
guides:salt/salt-yomi.adoc[].



.Procedure: Configuring the Yomi Formula
. Navigate to the menu:Formulas[Yomi] tab, and set these parameters in the
[guimenu]``General Configuration`` section:
* Check the [guimenu]``Events`` box to allow monitoring.
* In the [guimenu]``Reboot`` field, select ``yes`` to instruct the client to reboot after
installation.
* Check the [guimenu]``Snapper`` box if you are using the btrfs file system on the client.
* In the [guimenu]``Locale`` field, select the region and encoding for systemd to use on
the client.
    For example: ``en_US.utf8`` for US English and UTF-8.
* In the [guimenu]``Keymap`` field, select the appropriate keyboard layout.
    For example: ``us`` for a US keyboard layout.
* In the [guimenu]``Timezone`` field, select the timezone for the client to use.
    For example: [guimenu]``America/New_York`` for EST.
* In the [guimenu]``Hostname`` field, enter the hostname for the client to use.
    Leave this blank if you are using DHCP to provide the hostname.
* In the [guimenu]``Machine Id`` field, enter a machine identification number for the
client.
    Leave this blank to have systemd generate one automatically.
* In the [guimenu]``Target`` field, enter a systemd target unit.
. Click btn:[Save Formula] to save your configuration.



.Procedure: Configuring the Yomi Storage Formula
. Navigate to the menu:Formulas[Yomi Storage] tab, and set these parameters in the
menu:Partitions[Config] section:
* In the [guimenu]``Labels`` field, select the default partition table type to use.
* In the [guimenu]``Initial Gap`` field, select the default amount of space to leave
before the first partition.
    For example: ``1{nbsp}MB``, or use ``0`` to leave no space between partitions.
. For each device that you want to configure, in the menu:Partitions[Devices] section,
click btn:[+], and set these parameters:
* In the [guimenu]``Device`` field, type the mount point for the device.
    For example, ``/dev/sda``.
* In the [guimenu]``Label`` field, select the partition table type to use, if it is
different from the default label you selected.
* In the [guimenu]``Initial Gap`` field, select the amount of space to leave before the
first partition, if it is different from the default space you specified.
. For each partition that you want to create, in the menu:Partitions[Devices > Partitions]
section, click btn:[+], and set these parameters:
* In the [guimenu]``Partition Number`` field, enter a number for the partition.
    The number you enter here is appended to the device name to identify the partition.
For example, partition number ``1`` on device ``/dev/sda`` can be identified as
``/dev/sda1``.
* In the [guimenu]``Partition Name`` field, enter a name for the partition.
    Leave this blank if you have entered a partition number in the previous field.
* In the [guimenu]``Partition Size`` field, enter a size for the partition.
    For example: ``500{nbsp}MB``. Use ``rest`` to use all the remaining free space.
. For each file system that you want to create, in the [guimenu]``Filesystems`` section,
click btn:[+], and set these parameters:
* In the [guimenu]``Partition`` field, select the partition to create the file system on.
    For example, ``/dev/sda1``.
* In the [guimenu]``Filesystem`` field, select the file system type to create.
* In the [guimenu]``Mountpoint`` field, type the mount point for the file system.
    For example: ``/`` for root.
. Click btn:[Save Formula] to save your configuration.
```

```
[NOTE]
====
If you want to use LVM or RAID on your devices, click btn:[+] in the appropriate sections,
and complete the details for your environment.
====



.Procedure: Configuring the Yomi Bootloader Formula
. Navigate to the menu:Formulas[Yomi Bootloader] tab, and set these parameters in the
[guimenu]``Bootloader`` section:
* In the [guimenu]``Device`` field, type the path for the bootloader.
    For example, ``/dev/sda``.
* In the [guimenu]``Timeout`` field, select the number of seconds grub will wait before
booting the default menu entry.
* In the [guimenu]``Kernel`` field, type any additional kernel parameters you want to use.
    Any kernel parameters you add here will be appended to the
``GRUB_CMDLINE_LINUX_DEFAULT`` line during boot.
* In the [guimenu]``Terminal`` field, type the terminal to use for both terminal input and
output.
* In the [guimenu]``Serial Command`` field, type parameters for using the serial port.
    Use this only if you are using the serial console as the default terminal.
* In the [guimenu]``Gfxmode`` field, type the resolution to use for the graphical
terminal.
    Use this only if you are using the graphical console as the default terminal.
* Check the [guimenu]``Theme`` box to use GRUB2 default branding package.
* Check the [guimenu]``Disable OS Prober`` box to disable using the OS prober to discover
other installed operating systems.
. Click btn:[Save Formula] to save your configuration.



.Procedure: Configuring the Yomi Software Formula
. Navigate to the menu:Formulas[Yomi Software] tab, and set these parameters in the
menu:Software[Configuration] section:
* Check the [guimenu]``Minimal`` box to use a minimal installation, which only installs
packages listed as ``Required``.
. For each repository that you want to add, in the menu:Software[Repositories] section,
click btn:[+], and set these parameters:
* In the [guimenu]``Repository Name`` field, type a name for the repository.
* In the [guimenu]``Repository URL`` field, type the location of the repository.
. To add packages from each repository, in the menu:Software[Packages] section, click
btn:[+], and set these parameters:
* In the menu:Software[Packages] field, type the names of the packages to install, or type
a pattern to search for the appropriate packages.
    For example, ``pattern:enhanced_base glibc-locale``, or ``kernel-default``.
. In the menu:Software[Image] section, set these parameters:
* In the [guimenu]``Image URL`` field, type the location of the operating system ISO image
to use.
* In the [guimenu]``Md5`` field, type the MD5 hash to use to verify the ISO.
. In the menu:SUSEConect[Config] section, set these parameters:
* In the [guimenu]``Registration Code`` field, type the registration code for the client
you are installing.
    You can obtain this code from {scc}.
* In the [guimenu]``Email`` field, type the administrator email address to use.
* In the [guimenu]``Url`` field, type the address of the registration server to use.
    For example, use ``https://scc.suse.com``, to register with {scc}.
* In the [guimenu]``Version`` field, type the version of the product you are registering.
. For each product that you want to register, in the menu:SUSEConnect[Products] section,
click btn:[+], and set these parameters:
* In the [guimenu]``Product`` field, type each product you want to register.
    For example, ``<product_name>/1.1/x86``, for version 1.1 with {x86} architecture.
* In the menu:SUSEConnect[Packages] field, type the names of the packages to install, or
type a pattern to search for the appropriate packages.
    For example, ``pattern:enhanced_base glibc-locale``, or ``kernel-default``.
```

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




.Procedure: Configuring the Yomi Services Formula
. Navigate to the menu:Formulas[Yomi Services] tab, and set these parameters:
* Check the [guimenu]``Install salt-minion`` box to install and configure the client as a
Salt client.
. For each service you want to enable, in the menu:Services[Enabled] section, click
btn:[+], and set these parameters:
* In the [guimenu]``Service`` field, type the name of the service to enable.
    For example, ``salt-minion``.
. For each service you want to disable, in the menu:Services[Disabled] section, click
btn:[+], and set these parameters:
* In the [guimenu]``Service`` field, type the name of the service to disable.
. Click btn:[Save Formula] to save your configuration.




.Procedure: Configuring the Yomi Users Formula
. Navigate to the menu:Formulas[Yomi Users] tab.
. For each user you want to create, in the [guimenu]``Users`` section, click btn:[+], and
set these parameters:
* In the [guimenu]``Username`` field, type the name of the new user.
* In the [guimenu]``Password Hash`` field, type the hashed version of the password to use.
. To add a certificate for each user, in the menu:Users[Certificates] section, click
btn:[+], and add the certificate to the [guimenu]``Certificate`` field.
. Click btn:[Save Formula] to save your configuration.


When you have completed and saved all the forms, apply the highstate.


For more information about using Yomi, see xref:specialized-guides:salt/salt-yomi.adoc[].

:leveloffset: 2
:leveloffset: +3

[[formulas-custom]]
= Custom Salt Formulas

This section contains information about writing custom formulas, including formulas with
forms. You can write your own custom formulas, and make them available to client systems
in the {productname} {webui}.

For information about the formulas provided by {productname}, see xref:specialized-
guides:salt/salt-formulas-by-product.adoc[].



== File Structure Overview

RPM-based formulas must be placed in a specific directory structure to ensure that they
work correctly. A formula contains two separate directories: [path]``states``, and
[path]``metadata``. Folders in these directories need to have exactly matching names.

The formula states directory contains anything necessary for a Salt state to work
independently. This includes [path]``.sls`` files, a [path]``map.jinja`` file and any
other required files. This directory should only be modified by RPMs and should not be
edited manually. For example, the [package]``locale-formula`` states directory is located
in:
```

/usr/share/susemanager/formulas/states/locale/

```
To create formulas with forms, the metadata directory contains a [path]``form.yml`` file.
The [path]``form.yml`` file defines the forms for {productname}. The metadata directory
also contains an optional [path]``metadata.yml`` file with additional information about
the formula. For example, the [package]``locale-formula`` metadata directory is located
in:
```

/usr/share/susemanager/formulas/metadata/locale/

```
If you have a custom formula that is not in an RPM, it must be in a state directory
configured as a Salt file root. Custom state formula data must be in:
```

/srv/salt/<custom-formula-name>/

```
Custom metadata information must be in:
```

/srv/formula_metadata/<custom-formula-name>/

```
All custom folders must contain a [path]``form.yml`` file. These files are detected as
form recipes and are applied to groups and systems from the {webui}:
```

/srv/formula_metadata/<custom-formula-name>/form.yml

```
[NOTE]
====
The Salt formula directory changed in {productname}{nbsp}4.0. The old directory location,
[path]``/usr/share/susemanager/formulas``, will continue to work for some time. You should
ensure that you update to the new directory location, [path]``/usr/share/salt-formulas/``
as soon as possible.
====



== Define Formula with Forms Data

{productname} requires a file called [path]``form.yml``, to describe how formula data
should look within the {webui}. The [path]``form.yml`` file is used by {productname} to
generate the desired formula with forms, with values editable by a user.

The file contains a list of editable attributes that start with a `$` sign. These
attributes are used to determine how to display the formula in the {productname} {webui}.

For example, the [path]``form.yml`` that is included with the [package]``locale-formula``
is in:
```

/usr/share/susemanager/formulas/metadata/locale/form.yml

```
Part of that file looks like this:
```

timezone: $type: group

```
name:
  $type: select
  $values: ["CET",
            "Etc/Zulu"
            ]
  $default: CET
```

```
  hardware_clock_set_to_utc:
    $type: boolean
    $default: True
...
```

```
All values that start with a `$` sign are annotations used to display the UI that users
interact with. These annotations are not part of pillar data itself and are handled as
metadata.


This section lists the available attributes:

$type::
The most important attribute is the `$type` attribute. It defines the type of the pillar
value and the form-field that is generated. The supported types are:

** `text`
** `password`
** `number`
** `url`
** `email`
** `date`
** `time`
** `datetime`
** `boolean`
** `color`
** `select`
** `group`
** `edit-group`
** `namespace`
** `hidden-group` (obsolete, renamed to ``namespace``)


[NOTE]
====
The text attribute is the default and does not need to be specified explicitly.
====


Many of these values are self-explanatory:

* The `text` type generates a simple text field
* The `password` type generates a password field
* The `color` type generates a color picker

The ``group``, ``edit-group``, and `namespace` (formerly ``hidden-group``) types do not
generate an editable field and are used to structure form and pillar data. All these types
support nesting.

The `group` and `namespace` types differ slightly. The `group` type generates a visible
border with a heading. The `namespace` type shows nothing visually, and is only used to
structure pillar data.

The `edit-group` type allows you to structure and restrict editable fields in a more
```

```
flexible way. The `edit-group` type is a collection of items of the same kind. Collections
can have these four shapes:

* List of primitive items
* List of dictionaries
* Dictionary of primitive items
* Dictionary of dictionaries

The size of each collection is variable. Users can add or remove elements.

For example, `edit-group` supports the `$minItems` and `$maxItems` attributes, which
simplifies complex and repeatable input structures. These, and also `itemName`, are
optional.


$default::
Allows you to specify a default value to be displayed. This default value will be used if
no other value is entered. In an `edit-group` it allows you to create initial members of
the group and populate them with specified data.

$optional::
This type is a Boolean attribute. If it is `true` and the field is empty in the form, then
this field will not be generated in the formula data and the generated dictionary will not
contain the field name key. If it is `false` and the field is empty, the formula data will
contain a `<field name>: null` entry.

$ifEmpty::
This type is used if the field is empty. This usually occurs because the user did not
provide a value. The `ifEmpty` type can only be used when `$optional` is `false` or not
defined. If `$optional` is `true`, then `$ifEmpty` is ignored. In this example, the `DP2`
string would be used if the user leaves the field empty:
+
```

displayName: $type: string $ifEmpty: DP2

```
$name::
Allows you to specify the name of a value that is shown in the form. If this value is not
set, the pillar name is used and capitalized without underscores and dashes. Reference it
in the same section with ``pass:c[${name}]``.

$help and $placeholder::
These attributes are used to give a user a better understanding of what the value should
be. The `$help` type defines the message a user sees when hovering over a field The
`$placeholder` type displays a gray placeholder text in the field

Use `$placeholder` only with text fields like text, password, email or date fields. Do not
add a placeholder if you also use `$default`, as it will hide the placeholder.

$key::
Applicable only if the `edit-group` has the shape of a dictionary. When the pillar data is
a dictionary, the `$key` attribute determines the key of an entry in the dictionary.
+
For example:
+
```

user_passwords: $type: edit-group $minItems: 1 $prototype: $key: $type: text $type: text $default: alice: secret-password bob: you-shall-not-pass

```
+
Pillar:
```

```
+
```

user_passwords: alice: secret-password bob: you-shall-not-pass

```
$minItems and $maxItems::
In an ``edit-group``, `$minItems` and `$maxItems` specifies the lowest and highest numbers
for the group.

$itemName::
In an ``edit-group``, `$itemName` defines a template for the name to be used for the
members of the group.

$prototype::
In an ``edit-group``, `$prototype` is mandatory and defines the default pre-filled values
for newly added members in the group.

$scope::
Specifies a hierarchy level at which a value may be edited. Possible values are
``system``, `group`, and ``readonly``.
+
The default value is `$scope: system`, allows values to be edited at group and system
levels. A value can be entered for each system but if no value is entered the system will
fall back to the group default.
+
The ``$scope: group`` option makes a value editable only for a group. On the system level
you will be able to see the value, but not edit it.
+
The `$scope: readonly` option makes a field read-only. It can be used to show data to the
user, but will not allow them to edit it. This option should be used in combination with
the ``$default`` attribute.

$visibleIf::
+
[NOTE]
====
Deprecated in favor of `$visible`.
====
+
Allows you to show a field or group if a simple condition is met. An example condition is:
+
```

some_group#another_group#my_checkbox == true

```
+
The left part of the condition is the path to another value, and groups are separated by
`#` signs. The middle section of the condition should be either `==` for a value to be
equal or `!=` for values that should be not equal. The last field in the statement can be
any value which a field should have or not have.
+
The field with this attribute associated with it will be shown only when the condition is
met. In this example the field will be shown only if `my_checkbox` is checked. The ability
to use conditional statements is not limited to check boxes. It may also be used to check
values of select-fields, text-fields, and similar.
+
A check box should be structured like this:
+
```

some_group: $type: group

```
another_group:
  $type: group
```

```
my_checkbox:
  $type: boolean
```

```
+
Relative paths can be specified using prefix dots. One dot indicates a sibling, two dots
indicate a parent, and so on. This is mostly useful for ``edit-group``.
+
```

some_group: $type: group

```
another_group:
  $type: group
```

```
my_checkbox:
  $type: boolean
```

```
my_text:
  $visibleIf: .my_checkbox == true
```

```
yet_another_group:
  $type: group
```

```
my_text2:
  $visibleIf: ..another_group#my_checkbox == true
```

```
+
If you use multiple groups with the attribute, you can allow a users to select an option
and show a completely different form, dependent upon the selected value.
+
Values from hidden fields can be merged into the pillar data and sent to the client. A
formula must check the condition again and use the appropriate data. For example:
+
```

show_option: $type: checkbox some_text: $visibleIf: .show_option == true

```
+
```

{% if pillar.show_option %} do_something: with: {{ pillar.some_text }} {% endif %}

```
$values::
Can only be used together with ``$type`` Use to specify the different options in the
select-field. `$values` must be a list of possible values to select. For example:
+
```

select_something: $type: select $values: ["option1", "option2"]

```
+
Or:
+
```

select_something: $type: select $values: - option1 - option2

```
$visible::
Allows you to show a field or group if a condition is met. You must use the
https://github.com/TomFrost/jexl[jexl] expression language to write the condition.
+
Example structure:
+
```

some_group: $type: group

```
another_group:
  $type: group
```

```
my_checkbox:
  $type: boolean
```

```
+
An example condition is:
+
```

formValues.some_group.another_group.my_checkbox == true

```
+
The field with this attribute will only show if the condition is met. In this example, the
field will show only if ``my_checkbox`` is checked. You can also choose other elements for
the conditional statement, such as select fields or text fields.
+

+
If you use multiple groups with the attribute, users can select an option that will show a
completely different form, depending on the selected value.
+
Values from hidden fields can be merged into the pillar data and sent to the client. A
formula must check the condition again and use the appropriate data. For example:
+
```

show_option: $type: checkbox some_text: $visible: this.parent.value.show_option == true

```
+
```

{% if pillar.show_option %} do_something: with: {{ pillar.some_text }} {% endif %}

```
$disabled::
Allows you to disable a field or group if a condition is met. You must use the
https://github.com/TomFrost/jexl[jexl] expression language to write the condition.
+
If specified at group level it will disable all fields in that group.
```

```
$required::
Fields with this attribute are mandatory. Supports using the
https://github.com/TomFrost/jexl[jexl] expresion language.

$match::
Allows using a regular expression to validate the content of a text field.
+
It supports the regular expression features existing in JavaScript.
+
Example:
+
```

```
hardware:
  $type: text
  $name: Hardware Type and Address
  $placeholder: Enter hardware-type hardware-address (for example, "ethernet
AA:BB:CC:DD:EE:FF")
  $help: Hardware Identifier - prefix is mandatory
  $match: "\\w+ [A-Z]{2}:[A-Z]{2}:[A-Z]{2}:[A-Z]{2}:[A-Z]{2}:[A-Z]{2}"
```

```
=== Expression language
You must use the https://github.com/TomFrost/jexl[jexl] expression language to write
conditions.

Given a structure like this:
```

some_group: $type: group

```
another_group:
  $type: group
```

```
my_checkbox:
  $type: boolean
```

```
An example condition is:
```

formValues.some_group.another_group.my_checkbox == true

```
Absolute paths must begin with ``formValues``.

Specify relative paths using ``this.parent.value`` to define the value of the parent.

You can also refer to the parent of the parent, with ``this.parent.parent.value``. This is
mostly useful for ``edit-group`` elements.

Example for relative paths:
```

some_group: $type: group

```
another_group:
  $type: group
```

```
my_checkbox:
  $type: boolean
```

```
my_text:
  $visible: this.parent.value.my_checkbox
```

```
yet_another_group:
  $type: group
```

```
my_text2:
  $visible: this.parent.parent.value.another_group.my_checkbox
```

```
.Example: Basic edit-group
```

partitions: $name: "Hard Disk Partitions" $type: "edit-group" $minItems: 1 $maxItems: 4 $itemName: "Partition ${name}" $prototype: name: $default: "New partition" mountpoint: $default: "/var" size: $type: "number" $name: "Size in GB" $default: - name: "Boot" mountpoint: "/boot" - name: "Root" mountpoint: "/" size: 5000

```
Click btn:[Add] to fill the form with the default values.

The formula is called [path]``hd-partitions`` and will appear as [guimenu]``Hd
Partitions`` in the {webui}.

image::formula-custom-harddisk-partitions.png[]

To remove the definition of a partition click the minus symbol in the title line of an
inner group.

When you are finished, click btn:[Save Formula].


.Example: Nested edit-group
```

users: $name: "Users" $type: edit-group $minItems: 2 $maxItems: 5 $prototype: name: $default: "username" password: $type: password groups: $type: edit-group $minItems: 1 $prototype: group_name: $type: text $default: - name: "root" groups: - group_name: "users" - group_name: "admins" - name: "admin" groups: - group_name: "users"

```
== Writing Salt Formulas

Salt formulas are pre-written Salt states. You can use Jinja to configure formulas with
pillar data.

Basic Jinja syntax is:
```

pillar.some.value

```
When you are sure a pillar exists, use this syntax:
```

salt['pillar.get']('some:value', 'default value')

```
You can also replace the `pillar` value with `grains`. For example, ``grains.some.value``.

Using data this way makes the formula configurable. In this example, a specified package
is installed in the ``package_name`` pillar:
```

install_a_package: pkg.installed: - name: {{ pillar.package_name }}

```
You can also use more complex constructs such as `if/else` and `for-loops` to provide
greater functionality:
```

{% if pillar.installSomething %} something: pkg.installed {% else %} anotherPackage: pkg.installed {% endif %}

```
Another example:
```

{% for service in pillar.services %} start_{{ service }}: service.running: - name: {{ service }} {% endfor %}

```
Jinja also provides other helpful functions. For example, you can iterate over a
dictionary:
```

{% for key, value in some_dictionary.items() %} do_something_with_{{ key }}: {{ value }} {% endfor %}

```
You can have Salt manage your files (for example, configuration files for a program), and
change them with pillar data.

In this example, Salt copies the file [path]``salt-
file_roots/my_state/files/my_program.conf`` on the server to
[path]``/etc/my_program/my_program.conf`` on the client and template it with Jinja:
```

/etc/my_program/my_program.conf: file.managed: - source: salt://my_state/files/my_program.conf - template: jinja

```
This example allows you to use Jinja in the file, like the previous example for states:
```

some_config_option = {{ pillar.config_option_a }}

```
== Separate Data

Separating data from a state can increase flexibility and make it easier to re-use. You
can do this by writing values into a separate file named [path]``map.jinja``. This file
must be within the same directory as the state files.

This example sets `data` to a dictionary with different values, depending on which system
the state runs on. It will also merge data with the pillar using the `some.pillar.data`
value so you can access `some.pillar.data.value` by using ``data.value``.

You can choose to override defined values from pillars. For example, by overriding
`some.pillar.data.package` in this example:
```

{% set data = salt['grains.filter_by']({ 'Suse': { 'package': 'packageA', 'service': 'serviceA' }, 'RedHat': { 'package': 'package_a', 'service': 'service_a' } }, merge=salt['pillar.get']('some:pillar:data')) %}

```
When you have created a map file, you can maintain compatibility with multiple system
types while accessing deep pillar data in a simpler way.

Now you can import and use `data` in any file. For example:
```

{% from "some_folder/map.jinja" import data with context %}

install_package_a: pkg.installed: - name: {{ data.package }}

```
You can define multiple variables by copying the `{% set ...%}` statement with different
values and then merge it with other pillars. For example:
```

{% set server = salt['grains.filter_by']({ 'Suse': { 'package': 'my-server-pkg' } }, merge=salt['pillar.get']('myFormula:server')) %} {% set client = salt['grains.filter_by']({ 'Suse': { 'package': 'my-client-pkg' } }, merge=salt['pillar.get']('myFormula:client')) %}

```
To import multiple variables, separate them with a comma. For example:
```

{% from "map.jinja" import server, client with context %}

```
For more information about conventions to use when writing formulas, see
https://docs.saltproject.io/en/latest/topics/development/conventions/formulas.html.



== Generated Pillar Data

Pillar data is generated by {productname} when events occur like generating the highstate.
You can use an external pillar script to generate pillar data for packages and group IDs,
and include all pillar data for a system:
```

/usr/share/susemanager/modules/pillar/suma_minion.py

```
The process is executed like this:

. The `suma_minion.py` script starts and finds all formulas for a system by checking the
`group_formulas.json` and `server_formulas.json` files.
. The script loads the values for each formula (groups and from the system) and merges
them with the highstate.
    By default, if no values are found, a group overrides a system if `$scope: group`.
. The script also includes a list of formulas applied to the system in a pillar named
``formulas``.

This structure makes it possible to include states. In this example, the top file is
specifically generated by the `mgr_master_tops.py` script. The top file includes a state
called ``formulas`` for each system. This includes the `formulas.sls` file located in
[path]``/usr/share/susemanager/formulas/states`` or [path]``/usr/share/salt-
formulas/states/``. The content looks similar to this:
```

include: {{ pillar["formulas"] }}

```
This pillar includes all formulas that are specified in the pillar data generated from the
external pillar script.


Formulas should be created directly after {productname} is installed. If you encounter any
problems with formulas check these things first:

* The external pillar script (``suma_minion.py``) must include formula data.
* Data is saved to [path]``/srv/susemanager/formula_data`` and the [path]``pillar`` and
[path]``group_pillar`` sub-directories.
    These directories should be automatically generated by the server.
* Formulas must be included for every client listed in the top file.
    Currently this process is initiated by the [path]``mgr_master_tops.py`` script which
includes the `formulas.sls` file located in
[path]``/usr/share/susemanager/formulas/states/`` or [path]``/usr/share/salt-
formulas/states/``. This directory must be a salt file root. File roots are configured on
the salt-master ({productname}) located at [path]``/etc/salt/master.d/susemanager.conf``.

:leveloffset: 2
:leveloffset: +2

[[salt.ssh]]
= Salt SSH


Salt SSH allows Salt commands and states to be issued directly over SSH. SSH connections
are created on demand, when the server executes an action on a client.

For more information about Salt SSH, see
https://docs.saltproject.io/en/latest/topics/ssh/[https://docs.saltproject.io/en/latest/to
pics/ssh/].




== SSH Connection Methods

In {productname} there are two SSH connection methods, `ssh-push` and `ssh-push-tunnel`.
In both methods the server initiates an SSH connection to the client to execute a Salt
call.

In the `ssh-push` method, the package manager works as normal, and the HTTP or HTTPS
connection is directly created.

In the `ssh-push-tunnel` method, the server creates an HTTP or HTTPS connection through an
SSH tunnel. The HTTP connection initiated by the package manager is redirected through the
tunnel using `/etc/hosts` aliasing. Use this method for in-place firewall environments
that block HTTP or HTTPS connections between server and client.




== Salt SSH Integration

As with all Salt calls, {productname} invokes `salt-ssh` via the `salt-api`.

Salt SSH relies on a roster to obtain details such as hostname, ports, and the SSH
parameters of a client. {productname} keeps these details in the database and makes them
available to Salt SSH by using the [literal]``uyuni`` roster module, or by generating a
temporary roster file on bootstrapping new clients with the {webui}. The location of the
temporary roster file is supplied to Salt SSH using the [option]``roster-file`` option.
For the registered clients [option]``roster`` set to `uyuni` is used instead to get the
roster from the database with `uyuni` salt roster module.

[IMPORTANT]
====
It is not recommended to run [command]``salt-ssh`` as [literal]``root`` user. This can
cause permission issues the next time {productname} tries to use Salt SSH. Use
[command]``mgr-salt-ssh``, which changes the effective user to [literal]``salt`` and
avoids file permission issues. If you want to use [command]``mgr-salt-ssh`` with a user
other than [literal]``root``, the user should have the permission to change effective user
to [literal]``salt``.
====

[command]``mgr-salt-ssh`` uses [option]``roster`` set to [literal]``uyuni`` by default, if
neither [option]``roster`` nor [option]``roster-file`` specified in the command line. It
helps to call Salt commands to the registered Salt SSH clients with no roster file
generation.


== Authentication

Salt SSH supports both password and key authentication. {productname} uses both methods:

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

All other common Salt calls use key authentication. During the bootstrap step the SSH key
of the server is authorized on the client and added to the client
[path]``/.ssh/authorized_keys`` file. Subsequent calls no longer require a password.



[[salt.ssh.user]]
== User Account

The user for Salt SSH calls made by {productname} is taken from the `ssh_push_sudo_user`
setting. By default, the user is root.

[NOTE]
====
If bootstrapping with default settings fail, check whether the client allows root login
with ssh.
====

If the value of `ssh_push_sudo_user` is not root, then the `--sudo` options of `salt-ssh`
are used. For this user you must configure the `NOPASSWD` option in the [path]``sudoers``
file. At least, set the python binary with the version number; for example:
```

<USER> ALL=(ALL) NOPASSWD:/usr/bin/python3.6

```
== HTTP Redirection

The `ssh-push-tunnel` method requires traffic to be redirected through an SSH tunnel. This
allows traffic to bypass firewalls blocking a direct connection between the client and the
server.

This is achieved by using port 1233 in the repository URL:
```

<a href="https://suma-server:1233/repourl&#8230;&#8203" class="bare">https://suma-server:1233/repourl&#8230;&#8203</a>;

```
You can alias the suma-server hostname to [literal]``localhost`` in [path]``/etc/hosts``:
```

127.0.0.1 localhost suma-server

```
The server creates a reverse SSH tunnel that connects `localhost:1233` on the client to
`suma-server:443`:
```

ssh … -R 1233:suma-server:443

```
This means that the package manager will actually connect to `localhost:1233`, which is
then forwarded to `suma-server:443` by the SSH tunnel.

The package manager can contact the server only if the tunnel is open, which occurs only
when the server executes an action on the client.

Manual package manager operations that require server connectivity are not possible in
this case.



== Call Sequence

Salt SSH calls run in this sequence:

. Set `roster` parameter to `uyuni` for registered clients or prepare the Salt roster on
bootstrapping for the call
.. Create remote port forwarding option if the contact method is `ssh-push-tunnel`
.. Compute the `ProxyCommand` if the client is connected through a proxy
.. Create Roster content
. Create a temporary roster file (only in case of bootstrapping new client)
. Execute a synchronous `salt-ssh` call using the API
. Remove the temporary roster file (only in case of bootstrapping new client)

The roster content contains:

* `hostname`
* `user`
* `port`
* `remote_port_forwards`: The remote port forwarding SSH option
* `ssh_options`: Other ssh options:
** `ProxyCommand`: If the client connects through a proxy
* `timeout`: defaults to 180 seconds
* `minion_opts`:
** `master`: Set to the minion ID if the contact method is `ssh-push-tunnel`
* `ssh_pre_flight`: The path to the shell script executed on the client before running any
Salt command
* `ssh_pre_flight_args`: The list of arguments to call the pre flight script on the client



== Bootstrap Sequence

This section describes the sequence of events when clients are registered to a Salt
master. While bootstrapping is a type of Salt SSH call, the sequence differs slightly from
regular SSH calls.

Bootstrapping uses Salt SSH for communication between the master and the client. This
happens for both regular and SSH clients.



. For a regular Salt client, generate and pre-authorize the Salt key of the client.
. For an SSH client, if a proxy was selected, retrieve the SSH public key of the proxy
```

```
using the ``mgrutil.chain_ssh_cmd`` runner.
    The runner copies the public key of the proxy to the server using SSH. If needed, it
can chain multiple SSH commands to reach the proxy across multiple hops.
. Generate pillar data for bootstrap.
    The pillar data is compiled and stored on the Salt master, and retrieved by the
client.
. Generate the roster for bootstrapping into a temporary file on the client.
    You can use the roster by passing it to the Salt API, with this command:
+
```

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

```
+
For `bootstrap_state`, use `bootstrap` for regular clients or `ssh_bootstrap` for SSH
clients.


The way the client retrieves the pillar data depends on the contact method you have chosen
for your client:

* If you are using the `ssh-push-tunnel` contact method, ensure you have completed the
remote port forwarding option.
* If the client connects through a proxy, ensure you have completed the `ProxyCommand`
option.
    This depends on your proxy configuration, including how many proxies you need to
connect through.


Pillar data contains:

* mgr_server: The hostname of the Salt master
* mgr_origin_server: The hostname of the {productname} Server
* minion_id: The hostname of the client to bootstrap
* contact_method: The connection type
* mgr_sudo_user: The user for `salt-ssh`
* activation_key: If selected
* minion_pub: The pre-authorized public client key
* minion_pem: The pre-authorized private client key
* proxy_pub_key: The public SSH key that was retrieved from the proxy if the target is an
SSH client and a proxy was selected

The roster content contains:

* `hostname`
* `user`
* `password`
* `port`
* `remote_port_forwards`: the remote port forwarding SSH option
* `ssh_options`: other SSH options:
** `ProxyCommand` if the client connects through a proxy
* `timeout`: defaults to 180 seconds
* `ssh_pre_flight`: The path to the pre flight shell script (default:
[path]``/usr/share/susemanager/salt-ssh/preflight.sh``)
* `ssh_pre_flight_args`: The list of arguments to call the pre flight script on the client

This image provides an overview of the Salt SSH bootstrap process.

.Salt SSH Bootstrap Process


image::salt-ssh-bootstrap-process.png[]
// This image needs some exposition, at the very least. --LKB 2020-06-16
```

```
== Proxy Support

Salt SSH works with {productname} Proxy by chaining the SSH connection from one server or
proxy to the next. This is also known as a multi-hop or multi-gateway SSH connection.

image::salt-ssh-proxy-multi-hop.png[]


{productname} uses `ProxyCommand` to redirect SSH connections through proxies. This
options invokes an arbitrary command that is expected to connect to the SSH port on the
target host. The SSH process uses standard input and output of the command to communicate
with the remote SSH daemon.

`ProxyCommand` replaces a TCP/IP connection. It does not perform any authorization or
encryption. Its role is simply to create a byte stream to the remote SSH daemon port.

This image depicts a client connecting to a server that is behind a gateway. In this
example `netcat` is used to pipe port 22 of the target host into the SSH standard
input/output:

image::salt-ssh-proxycommand.png[]


The Salt SSH calls run in this sequence when a proxy is in use:

. {productname} initiates the SSH connection.
. `ProxyCommand` uses SSH to create a connection from the server to the client through the
proxies.


This example uses `ProxyCommand` with two proxies and the `ssh-push` method:
```

Connect the server to the first proxy:

/usr/bin/ssh -i /srv/susemanager/salt/salt_ssh/mgr_ssh_id -o StrictHostKeyChecking=no -o User=mgrsshtunnel proxy1

Connect the first proxy to the second, and forward standard input/output on the client to client:22 using the `-W` option:

/usr/bin/ssh -i /var/lib/spacewalk/mgrsshtunnel/.ssh/id_susemanager_ssh_push -o StrictHostKeyChecking=no -o User=mgrsshtunnel -W client:22 proxy2

```
image::salt-ssh-push-push-plain-sequence.png[]

This example uses `ProxyCommand` with two proxies and the `ssh-push-tunnel` method:
```

Connect the server to the first proxy:

/usr/bin/ssh -i /srv/susemanager/salt/salt_ssh/mgr_ssh_id -o User=mgrsshtunnel proxy1

Connect the first proxy to the second:

/usr/bin/ssh -i /home/mgrsshtunnel/.ssh/id_susemanager_ssh_push -o User=mgrsshtunnel proxy2

Connect the second proxy to the client and open an reverse tunnel (-R 1233:proxy2:443) from the client to the HTTPS port on the second proxy:

/usr/bin/ssh -i /home/mgrsshtunnel/.ssh/id_susemanager_ssh_push -o User=root -R 1233:proxy2:443 client

Connect the client to itself and forward the standard input/output of the server to the SSH port of the client (-W client:22).

This is equivalent to **ssh … proxy2 netcat client 22`** and is needed because SSH does not allow both the reverse tunnel (-R 1233:proxy2:443) and the standard input/output forward (-W client:22) in the same command. /usr/bin/ssh -i /root/.ssh/mgr_own_id -W client:22 -o User=root client

```
image::salt-ssh-push-push-tunnel-sequence.png[]



== Users and SSH Key Management

To connect to a proxy, the parent server or proxy uses a specific user called
`mgrsshtunnel`. When `mgrsshtunnel` connects, the SSH configuration of the proxy will
force the execution of ``/usr/sbin/mgr-proxy-ssh-force-cmd``. This is a simple shell
script that allows only the execution of `scp`, `ssh`, or `cat` commands.

The connection to the proxy or client is authorized using SSH keys in this sequence:

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

image::salt-ssh-push-ssh-keys.png[]



== Repository Access with a Proxy

When {productname} connects to a repository using a proxy, it can use either `ssh-push` or
`ssh-push-tunnel`.

In both methods the client connects to the proxy to retrieve package and repository
information.

In the `ssh-push` method, the package manager connects directly to the proxy using HTTP or
HTTPS. This works in cases where there is no firewall between the client and the proxy
that blocks HTTP connections initiated by the client.

image::salt-ssh-push-repo-access.png[]
```

```
In the `ssh-push-tunnel` method, the HTTP connection to the proxy is redirected through a
reverse SSH tunnel.

image::salt-ssh-push-tunnel-repo-access.png[]



== Proxy Setup

When the `spacewalk-proxy` package is installed on the proxy, the `mgrsshtunnel` user is
created.

The initial configuration with ``configure-proxy.sh`` occurs using this sequence:

. An SSH key pair is generated, or an existing keypair is imported.
. The SSH key of the parent server or proxy is retrieved to authorize it on the proxy.
. The ``ssh`` daemon on the proxy is configured to restrict the ``mgrsshtunnel`` user.
    This is done by the ``mgr-proxy-ssh-push-init`` script, which is called from
``configure-proxy.sh``. It does not have to be manually invoked.

The parent key is retrieved by calling an HTTPS endpoint on the parent server or proxy.
The first endpoint tried is **+https://$PARENT/pub/id_susemanager_ssh_push.pub+**. If the
parent is a proxy then this will return the public SSH key of the proxy.

If a 404 error is received from that endpoint, then the parent is assumed to be a server
not a proxy, and **+https://$PARENT/rhn/manager/download/saltssh/pubkey+** is tried
instead.

If an SSH key exists at ``/srv/susemanager/salt/salt_ssh/mgr_ssh_id.pub`` on the server it
is returned.

If the public key does not exist because ``salt-ssh`` has not been invoked yet, a key will
be generates by calling the ``mgrutil.ssh_keygen`` runner.

[NOTE]
====
Salt SSH generates a keypair the first time it is invoked with
``/srv/susemanager/salt/salt_ssh/mgr_ssh_id``. The sequence in this section is needed if a
proxy is configured before Salt SSH was invoked for the first time.
====



[[salt.ssh.key_rotation]]
== Rotate SSH keys

The SSH key is used on [literal]``salt-ssh`` managed systems. Additionally, it is used on
{productname} Proxies for the user [systemitem]``mgrsshtunnel``. Normal systems managed
with the {salt} [literal]``default`` method are not affected and do not have this key
configured.

[NOTE]
====
Environments without SSH managed systems may not have the key at all. Traditionally
managed systems with SSH push are not affected and the key cannot be rotated by this
mechanism.
====



.Procedure: Rotating SSH keys

. On the {productname} Server, as user [systemitem]``root``, change to user
[systemitem]``salt``:
+
```

su -s /bin/bash - salt

```
. Create a new SSH key:
+
```

ssh-keygen -N "" -t rsa -q -f /var/lib/salt/.ssh/new_mgr_ssh_id

```
. Copy the public key into the {salt} filesystem to make it usable in a {salt} state:
+
```

cp /var/lib/salt/.ssh/new_mgr_ssh_id.pub /srv/susemanager/salt/salt_ssh/

```
. Change back to user [systemitem]``root`` again:
+
```

exit

1. Rollout the new key to all systems that need it. Applying the **util.mgr_rotate_saltssh_key** state will limit the changes to **salt-ssh** managed systems and proxies:

   ```
   salt '*' state.apply util.mgr_rotate_saltssh_key
   mgr-salt-ssh '*' state.apply util.mgr_rotate_saltssh_key
   ```

2. Move the old key away and make the new key the standard key. Rename **mgr_ssh_id** key to **disabled_mgr_ssh_id** and **new_mgr_ssh_id** key to **mgr_ssh_id** in the SSH keystore of the user **salt** as well as in the Salt filesystem for the public keys:

   ```
   su -s /bin/bash - salt
   cd .ssh
   mv mgr_ssh_id disabled_mgr_ssh_id
   mv mgr_ssh_id.pub disabled_mgr_ssh_id.pub
   mv new_mgr_ssh_id mgr_ssh_id
   mv new_mgr_ssh_id.pub mgr_ssh_id.pub
   cd /srv/susemanager/salt/salt_ssh/
   mv mgr_ssh_id.pub disabled_mgr_ssh_id.pub
   mv new_mgr_ssh_id.pub mgr_ssh_id.pub
   ```

3. OPTIONAL: When containerized proxies exist, re-create the configuration to get the new SSH key into the proxy configuration. Restart the containers with the new configuration. It is also possible to change the existing configuration on the podman host (**ssh.yaml**). Change the value of **server_ssh_key_pub** with the content of the current **mgr_ssh_id.pub**.

4. To remove the disabled keys from the **authorized_keys` files of the [literal]**salt-ssh`` managed systems and the proxies, apply the state a second time:

   ```
   salt '*' state.apply util.mgr_rotate_saltssh_key
   mgr-salt-ssh '*' state.apply util.mgr_rotate_saltssh_key
   ```

Salt Rate Limiting

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

These parameters are all configured in the **/etc/rhn/rhn.conf** configuration file.

Salt commands that are executed from the command line are not subject to these parameters.

Batching

There are two parameters that control how actions are sent to clients, one for the batch size, and one for the delay.

When the SUSE Manager Server sends a batch of actions to the target clients, it will send it to the number of clients determined in the batch size parameter. After the specified delay period, commands will be sent to the next batch of clients. The number of clients in each subsequent batch is equal to the number of clients that have completed in the previous batch.

Choosing a lower batch size will reduce system load and parallelism, but might reduce overall performance for processing actions.

The batch size parameter sets the maximum number of clients that can execute a single action at the same time. Adjust the **java.salt_batch_size** parameter. Defaults to 200.

Increasing the delay increases the chance that multiple clients will have completed before the next action is issued (more clients are grouped together in subsequent batches), resulting in fewer overall commands, and reducing load.

The batch delay parameter sets the amount of time, in seconds, to wait after a command from the previous batch is processed before beginning to process the command on the next client. Adjust the **java.salt_batch_delay** parameter. Defaults to 1.0 seconds.

Disabling the Salt Mine

In older versions, SUSE Manager used a tool called Salt mine to check client availability. The Salt mine would cause clients to contact the server every hour, which created significant load. With the introduction of a more efficient mechanism in SUSE Manager 3.2, the Salt mine is no longer required. Instead, the SUSE Manager Server uses Taskomatic to ping only the clients that appear to have been offline for twelve hours or more, with all clients being contacted at least once in every twenty four hour period by default. You can adjust this by changing the **web.system_checkin_threshold** parameter in **rhn.conf**. The value is expressed in days, and the default value is **1**.

Newly registered Salt clients will have the Salt mine disabled by default. If the Salt mine is running on your system, you can reduce load by disabling it. This is especially effective if you have a large number of clients.

Disable the Salt mine by running this command on the server:

```
salt '*' state.sls util.mgr_mine_config_clean_up
```

This will restart the clients and generate some Salt events to be processed by the server. If you have a large number of clients, handling these events could create excessive load. To avoid this, you can execute the command in batch mode with this command:

```
salt --batch-size 50 '*' state.sls util.mgr_mine_config_clean_up
```

You will need to wait for this command to finish executing. Do not end the process with `Ctrl` + `C`.

Scaling Minions (Large Scale Deployments)

SUSE Manager is designed by default to work on small and medium scale installations. For installations with more than 1000 clients per SUSE Manager Server, adequate hardware sizing and parameter tuning must be performed.

For more information on managing large scale deployments, see **Specialized-guides › Large-deployments**.

Salt Connectivity

Depending on the size and spreadness of the environment where SUSE Manager is used some connectivity issues are possible. There are no common recommendations for all the possible use cases as the environments could be very different expecially if public clouds instances are involved.

Minions Connectivity

In case clients are losing the connection to the SUSE Manager Server (or SUSE Manager Proxy if involved), they are unreachable from the SUSE Manager Server Web UI or with command line tools. To understand such a connection issue check if the client is not reachable from the SUSE Manager Server with **salt MINION_ID test.ping**, while **venv-salt-call test.ping** or (**salt-call test.ping** in case if non-bundle salt is used on the client) is working fine on the client side. If this is the case, it is recommended to set **tcp_keepalive** parameters.

The parameters with the values from the example below could be used as a starting point to look for the better combination, which could prevent cases of connection loss without restoring for some environments. It is recommended to put the parameters in the separate drop-in configuration file like **/etc/venv-salt-minion/minion.d/tuning-keepalives.conf** or **/etc/salt/minion.d/tuning-keepalives.conf**, depending on the minion type used on the client side.

列表 2. Example: Keepalive parameters example values

```
######      Keepalive settings        ######
############################################
# ZeroMQ now includes support for configuring SO_KEEPALIVE if supported by
# the OS. If connections between the minion and the master pass through
# a state tracking device such as a firewall or VPN gateway, there is
# the risk that it could tear down the connection the master and minion
# without informing either party that their connection has been taken away.
# Enabling TCP Keepalives prevents this from happening.
# Overall state of TCP Keepalives, enable (1 or True), disable (0 or False)
# or leave to the OS defaults (-1), on Linux, typically disabled. Default True, enabled.
tcp_keepalive: True

# How long before the first keepalive should be sent in seconds. Default 300
# to send the first keepalive after 5 minutes, OS default (-1) is typically 7200 seconds
# on Linux see /proc/sys/net/ipv4/tcp_keepalive_time.
tcp_keepalive_idle: 10

# How many lost probes are needed to consider the connection lost. Default -1
# to use OS defaults, typically 9 on Linux, see /proc/sys/net/ipv4/tcp_keepalive_probes.
tcp_keepalive_cnt: 3

# How often, in seconds, to send keepalives after the first one. Default -1 to
# use OS defaults, typically 75 seconds on Linux, see
# /proc/sys/net/ipv4/tcp_keepalive_intvl.
tcp_keepalive_intvl: 10
```

Proxies Connectivity

salt-broker service is used on SUSE Manager Proxies to forward **salt** traffic between the SUSE Manager Server and **salt-minion** service used on the client side. It is possible that **salt-broker** and all the clients behind it could be affected with the same issue as the clients directly connected to the SUSE Manager Server. The issue could be fixed with the same parameters as recommended for the minions, but specified in **/etc/salt/broker** on each SUSE Manager Proxy.

The other possible issue which SUSE Manager Proxy can be affected with is the case when the connectivity to the SUSE Manager Server is lost for quite long interval, so the clients behind it started to retry the authentication to the **salt-master** service on the SUSE Manager Server. This situation could be potentially dangerous as it could lead to collecting large amount of ZeroMQ messages with authentication attemps in the interal buffer of ZeroMQ sockets used inside the **salt-broker** service, so that on restoring the connection to the **salt-master** all of the messages will be pushed to it. It could lead to the issues on SUSE Manager Server side with **salt-master** service, as it could be impossible to serve all the cached requests in apropriate time or even to the complete denial of the service.

To avoid such situation a set of extra parameters was introduced. The most important one is **wait_for_backend**, which should be set to **True**. This prevents opening the sockets for the clients behind the proxy while the connectivity to the **salt-master** service is not established. In this case the messages from the clients are not collected in the internal buffers. **drop_after_retries** is setting the number of retries before closing the sockets to drop the cached messages. The other parameters could help to fine tune the behaviour for the environment.

Setting timeouts, intervals and **drop_after_retries** to lower values are making **salt-broker** service more aggressive on detecting the connection loss to the **salt-master**, so that it puts the clients behind the proxy to the conditions closer as they are connected directly to the SUSE Manager Server. On the other hand increasing the values could provide some benefits in case if the network channel between the SUSE Manager Proxy and SUSE Manager Server is not stable enough, the message buffering can provide some flexibility while re-establishing the connection.

列表 3. Example: Keepalive parameters example values

```
######    ZeroMQ connection options    ######
############################################
```

```
# For more details about the following parameters check ZeroMQ documentation:
# http://api.zeromq.org/4-2:zmq-setsockopt
# All of these parameters will be set to the backend sockets
# (from the salt-broker to the salt-master)

# connect_timeout (sets ZMQ_CONNECT_TIMEOUT)
# default: 0
# value unit: milliseconds
# Sets how long to wait before timing-out a connect to the remote socket.
# 0 could take much time, so it could be better to set to more strict value
# for particular environment depending on the network conditions.
# The value equal to 10000 is setting 10 seconds connect timeout.
connect_timeout: 3000

# reconnect_ivl (sets ZMQ_RECONNECT_IVL)
# default: 100
# value unit: milliseconds
# Sets the interval of time before reconnection attempt on connection drop.
reconnect_ivl: 1000

# heartbeat_ivl (sets ZMQ_HEARTBEAT_IVL)
# default: 0
# value unit: milliseconds
# This parameter is important for detection of loosing the connection.
# In case of value equal to 0 it is not sending heartbits.
# It's better to set to more relevant value for the particular environment,
# depending on possible network issues.
# The value equal to 20000 (20 seconds) works good for most cases.
heartbeat_ivl: 5000

# heartbeat_timeout (sets ZMQ_HEARTBEAT_TIMEOUT)
# default: 0
# value unit: milliseconds
# Sets the interval of time to consider that the connection is timed out
# after sending the heartbeat and not getting the response on it.
# The value equal to 60000 (1 minute) is considering the connection is down
# after 1 minute of no response to the heartbeat.
heartbeat_timeout: 10000


######   Other connection options    ######
# The following parameters are not related to ZeroMQ,
# but the internal parameters of the salt-broker.
# drop_after_retries
# default: -1
# value unit: number of retries
# Drop the frontend sockets of the salt-broker in case if it reaches
# the number of retries to reconnect to the backend socket.
# -1 means not drop the frontend sockets
# It's better to choose more relevant value for the particular environment.
# 10 can be a good choise for most of the cases.
drop_after_retries: 5

# wait_for_backend
# default: False
# The main aim of this parameter is to prevent  collecting the messages
# with the open frontend socket and prevent pushing them on connecting
# the backend socket to prevent large number of messages to be pushed
# at once to salt-master.
# It's better to set it to True if there is significant numer of minions
# behind the salt-broker.
wait_for_backend: True
```

Salt timeouts

General Salt timeouts

Salt features two timeout parameters called **timeout** and **gather_job_timeout** that are relevant during the execution of Salt commands and jobs—it does not matter whether they are triggered using the command line interface or API. These two parameters are explained in the following article.

This is a normal workflow when all clients are well reachable:

- A salt command or job is executed:

```
salt '*' test.ping
```

- Salt master publishes the job with the targeted clients into the Salt PUB channel.
- Clients take that job and start working on it.
- Salt master is looking at the Salt RET channel to gather responses from the clients.
- If Salt master gets all responses from targeted clients, then everything is completed and Salt master will return a response containing all the client responses.

If some of the clients are down during this process, the workflow continues as follows:

1. If **timeout** is reached before getting all expected responses from the clients, then Salt master would trigger an aditional job (a Salt **find_job** job) targeting only pending clients to check whether the job is already running on the client.
2. Now **gather_job_timeout** is evaluated. A new counter is now triggered.
3. If this new **find_job** job responds that the original job is actually running on the client, then Salt master will wait for that client’ s response for another **gather_job_timeout** interval before issuing the next **find_job** job.
4. In case of reaching **gather_job_timeout** without having any response from the client (neither for the initial **test.ping** nor for the **find_job** job), Salt master will return with only the gathered responses from the responding clients.

By default, SUSE Manager globally sets **timeout** and **gather_job_timeout** to 120 seconds. So, in the worst case, a Salt call targeting unreachable clients will end up with 240 seconds of waiting until getting a response.

You can configure these values differently by creating a **/etc/salt/master.d/custom.conf** configuration file according to syntax in **/etc/salt/master.conf**.

Presence Ping Timeouts

Before Actions are executed on Salt clients, whether they scheduled via the Web UI or the API, SUSE Manager performs a "presence ping" command to ensure the respective **salt-minion** processes are active and able to respond. Then, a ping gather job runs on the Salt master to handle the incoming pings from the clients. Actual commands will begin only after all clients have either responded to the ping, or timed out.

The presence ping is an ordinary Salt command, but is not subject to the same timeout parameters as all other Salt commands (**timeout**/**gather_job_timeout**, described above). Rather, it has its own parameters (**java.salt_presence_ping_timeout**/**java.salt_presence_ping_gather_job_timeout**) that can be set in **/etc/rhn/rhn.conf**.

To allow for quicker detection of unresponsive clients, the timeout values for presence pings are by default significantly shorter than the general defaults. You can configure the presence ping parameters in **/etc/rhn/rhn.conf**, however the default values should be sufficient in most cases.

A lower total presence ping timeout value will increase the chance of false negatives. In some cases, a client might be marked as non-responding, when it is responding but did not respond quickly enough. Additionally, setting this total presence ping timeout value too low could result in a client hanging at the boot screen. A higher total presence ping timeout will increase the accuracy of the test, as even slow clients will respond to the presence ping before timing out. Additionally, a higher presence ping timeout could limit throughput if you are targeting a large number of clients, when some of them are slow.

If a client does not reply to a ping within the allocated time, it is marked as **not available**, and is excluded from the command. The Web UI shows a **minion is down or could not be contacted** message in this case.

The presence ping timeout parameter changes the timeout setting for the presence ping, in seconds. Adjust the **java.salt_presence_ping_timeout** parameter. Defaults to 4 seconds.

The presence ping gather job parameter changes the timeout setting for gathering the presence ping, in seconds. Adjust the **java.salt_presence_ping_gather_job_timeout** parameter. Defaults to 1 second.

Salt SSH Clients (SSH Push)

Salt SSH clients are slightly different than regular clients (zeromq). Salt SSH clients do not use Salt PUB/RET channels but a wrapper Salt command inside of an SSH call. Salt **timeout** and **gather_job_timeout** are not playing a role here.

SUSE Manager defines a timeout for SSH connections in **/etc/rhn/rhn.conf**:

```
# salt_ssh_connect_timeout = 180
```

Large Deployments Guide Overview

Updated: 2025-07-21

SUSE Manager is designed by default to work on small and medium scale installations. For installations with more than 1000 clients per SUSE Manager Server, adequate hardware sizing and parameter tuning must be performed.

There is no hard maximum number of supported systems. Many factors can affect how many clients can reliably be used in a particular installation. Factors can include which features are used, and how the hardware and systems are configured.

Large installations require standard Salt clients. These instructions cannot be used in environments using traditional clients or Salt SSH minions.

There are two main ways to manage large scale deployments. You can manage them with a single SUSE Manager Server, or you can use multiple servers in a hub. Both methods are described in this book.

With large scale environments one should also consider the usage of SUSE Manager Proxies. Sizing and location of the Proxies will dependent on the deployment topology. For more information see **Installation-and-upgrade › Install-proxy**.

Additionally, if you are operating within a Retail environment, you can use SUSE Manager for Retail to manage large deployments of point-of-service terminals. There is an introduction to SUSE Manager for Retail in this book.

Tuning and monitoring large scale deployments can differ from smaller installations. This book contains guidance for both tuning and monitoring within larger installations.

Hardware Requirements

Not all problems can be solved with better hardware, but choosing the right hardware is an absolute necessity for large scale deployments.

The minimum requirements for the SUSE Manager Server are:

- Eight or more recent x86-64 CPU cores.
- 32 GiB RAM. For installations with thousands of clients, use 64 GB or more.
- Fast I/O storage devices, such as locally attached SSDs. For PostgreSQL data directories, we recommend locally attached RAID-0 SSDs.

If the SUSE Manager Server is virtualized, enable the **elevator=none** kernel command line option, for the best input/output performance. You can check the current status with **cat /sys/block/<DEVICE>/queue/scheduler**. This command will display a list of available schedulers with the currently active one in brackets. To change the scheduler before a reboot, use **echo none > /sys/block/<DEVICE>/queue/scheduler**.

The minimum requirements for the SUSE Manager Proxy are:

- One SUSE Manager Proxy per 500-1000 clients, depending on available network bandwidth.
- Two or more recent x86-64 CPU cores.
- 16 GB RAM, and sufficient storage for caching.

Clients should never be directly attached to the SUSE Manager Server in production systems.

In large scale installations, the SUSE Manager Proxy is used primarily as a local cache for content between the server and clients. Using proxies in this way can substantially reduce download time for clients, and decrease Server egress bandwidth use.

The number of clients per proxy will affect the download time. Always take network structure and available bandwidth into account.

We recommend you estimate the download time of typical usage to determine how many clients to connnect to each proxy. To do this, you will need to estimate the number of package upgrades required in every patch cycle. You can use this formula to calculate the download time:

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

For example, the total time needed to transfer 400 MB of upgrades through a physical link speed of 1 GB/s to 3000 clients:

```
400 MB  * 3000 / 119 MB/s / 60 = 169 min
```

Using a Single Server to Manage Large Scale Deployments

This section discusses how to set up a single SUSE Manager Server to manage a large number of clients. It contains some recommendations for hardware and networking, and an overview of the tuning parameters that you need to consider in a large scale deployment.

Operation Recommendations

This section contains a range of recommendations for large scale deployments.

Always start small and scale up gradually. Monitor the server as you scale to identify problems early.

Salt Client Onboarding Rate

The rate at which SUSE Manager can onboard clients is limited and depends on hardware resources. Onboarding clients at a faster rate than SUSE Manager is configured for will build up a backlog of unprocessed keys. This slows down the process and can potentially exhaust resources. We recommend that you limit the acceptance key rate programmatically. A safe starting point would be to onboard a client every 15 seconds. You can do that with this command:

```
for k in $(salt-key -l un|grep -v Unaccepted); do salt-key -y -a $k; sleep 15; done
```

Salt Clients and the RNG

All communication to and from Salt clients is encrypted. During client onboarding, Salt uses asymmetric cryptography, which requires available entropy from the Random Number Generator (RNG) facility in the kernel. If sufficient entropy is not available from the RNG, it will significantly slow down communications. This is especially true in virtualized environments. Ensure enough entropy is present, or change the virtualization host options.

You can check the amount of available entropy with the **cat /proc/sys/kernel/random/entropy_avail**. It should never be below 100-200.

Clients Running with Unaccepted Salt Keys

Idle clients which have not been onboarded, that is clients running with unaccepted Salt keys, consume more resources than idle clients that have been onboarded. Generally, this consumes about an extra 2.5 Kb/s of inbound network bandwidth per client. For example, 1000 idle clients will consume about

2.5 Mb/s extra. This consumption will reduce almost to zero when onboarding has been completed for all clients. Limit the number of non-onboarded clients for optimal performance.

Disabling the Salt Mine

In older versions, SUSE Manager used a tool called Salt mine to check client availability. The Salt mine would cause clients to contact the server every hour, which created significant load. With the introduction of a more efficient mechanism in SUSE Manager 3.2, the Salt mine is no longer required. Instead, the SUSE Manager Server uses Taskomatic to ping only the clients that appear to have been offline for twelve hours or more, with all clients being contacted at least once in every twenty four hour period by default. You can adjust this by changing the **web.system_checkin_threshold** parameter in **rhn.conf**. The value is expressed in days, and the default value is **1**.

Newly registered Salt clients will have the Salt mine disabled by default. If the Salt mine is running on your system, you can reduce load by disabling it. This is especially effective if you have a large number of clients.

Disable the Salt mine by running this command on the server:

```
salt '*' state.sls util.mgr_mine_config_clean_up
```

This will restart the clients and generate some Salt events to be processed by the server. If you have a large number of clients, handling these events could create excessive load. To avoid this, you can execute the command in batch mode with this command:

```
salt --batch-size 50 '*' state.sls util.mgr_mine_config_clean_up
```

You will need to wait for this command to finish executing. Do not end the process with `Ctrl` + `C`.

Disable Unnecessary Taskomatic jobs

To minimize wasted resources, you can disable non-essential or unused Taskomatic jobs.

You can see the list of Taskomatic jobs in the SUSE Manager Web UI, at **Admin › Task Schedules**.

To disable a job, click the name of the job you want to disable, select **Disable Schedule**, and click **[Update Schedule]**.

To delete a job, click the name of the job you want to delete, and click **[Delete Schedule]**.

We recommend disabling these jobs:

- Daily comparison of configuration files: **compare-configs-default**
- Hourly synchronization of Cobbler files: **cobbler-sync-default**
- Daily gatherer and subscription matcher: **gatherer-matcher-default**

Do not attempt to disable any other jobs, as it could prevent SUSE Manager from functioning correctly.

Swap and Monitoring

It is especially important in large scale deployments that you keep your SUSE Manager Server constantly monitored and backed up.

Swap space use can have significant impacts on performance. If significant non-transient swap usage is detected, you can increase the available hardware RAM.

You can also consider tuning the Server to consume less memory. For more information on tuning, see **Specialized-guides › Salt**.

AES Key Rotation

Communications from the Salt Master to clients is encrypted with a single AES key. The key is rotated when:

- The **salt-master** process is restarted, or
- Any minion key is deleted (for example, when a client is deleted from SUSE Manager)

After the AES key has been rotated, all clients must re-authenticate to the master. By default, this happens next time a client receives a message. If you have a large number of clients (several thousands), this can cause a high CPU load on the SUSE Manager Server. If the CPU load is excessive, we recommend that you delete keys in batches, and in off-peak hours if possible, to avoid overloading the server.

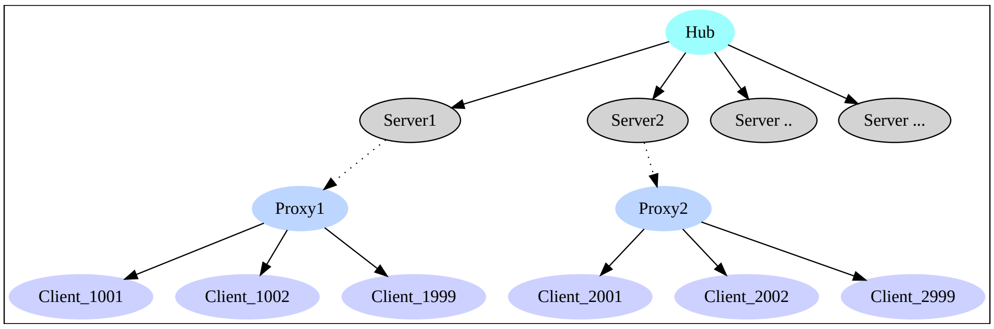
For more information, see:

- https://docs.saltproject.io/en/latest/topics/tutorials/intro_scale.html#too-many-minions-re-authing
- https://docs.saltproject.io/en/latest/ref/configuration/master.html#ping-on-rotate

Using Multiple Servers to Manage Large Scale Deployments

If you need to manage a large number of clients, in most cases you can do so with a single SUSE Manager Server, tuned appropriately. However, if you need to manage tens of thousands of clients, you might find it easier to use multiple SUSE Manager Servers, in a hub, to manage them.

SUSE Manager Hub helps you manage very large deployments. The typical Hub topology looks like this:



To copy contents between servers, you can use Inter-Server Synchronization (ISS). ISS allows you to export data from one server (source) and import it on another (target) server. For more information, see

Administration › Iss_intro.

Hub Requirements

To set up a Hub installation, you require:

- One central SUSE Manager Server, which acts as the Hub Server.
- One or more additional SUSE Manager Servers, registered to the Hub as Salt clients. This document refers to these as peripheral servers.
- Any number of clients registered to the peripheral servers.
- Ensure the Hub Server and all peripheral servers are running SUSE Manager 4.1 or higher.

The Hub Server must not have clients registered to it. Clients should only be registered to the peripheral servers. The exceptions are build hosts, which can be connected to the Hub.

Peripheral Servers

Peripheral servers must be registered to the Hub Server as Salt clients. When you register the peripheral servers, assign them the appropriate **SUSE Manager Server** software channel as their base channel. Additionally, they must be registered to the Hub Server directly, do not use a proxy.

For more information about registering clients, see **Client-configuration › Registration-webui**.

You need credentials to access the XMLRPC APIs on each server, including the Hub Server.

Hub Installation

Before you begin, you need to install the **hub-xmlrpc-api** package, and configure the Hub Server to use the API.

Procedure: Installing and Configuring the Hub XMLRPC API

1. On the Hub Server, or on a host that has access to all peripheral servers' XMLRPC APIs, install the **hub-xmlrpc-api** package. The package is available in the SUSE Manager 4.3 repositories.
2. OPTIONAL: Set the Hub XMLRPC API service to start automatically at boot time, and start it immediately:

```
sudo systemctl enable hub-xmlrpc-api.service
sudo systemctl start hub-xmlrpc-api.service
```

3. OPTIONAL: Check that these parameters in the **/etc/hub/hub.conf** configuration file are correct:
   - **HUB_API_URL**: URL to the Hub Server XMLRPC API endpoint. Use the default value if you are installing **hub-xmlrpc-api** on the Hub Server.
   - **HUB_CONNECT_TIMEOUT**: the maximum number of seconds to wait for a response when connecting to a Server. Use the default value in most cases.

- **HUB_REQUEST_TIMEOUT**: the maximum number of seconds to wait for a response when calling a Server method. Use the default value in most cases.
- **HUB_CONNECT_USING_SSL**: use HTTPS instead of HTTP for communicating with peripheral Servers. Recommended for a secure environment.

4. Restart services to pick up configuration changes.

> To use HTTPS to connect to peripheral Servers, you must set the **HUB_CONNECT_USING_SSL** parameter to **true**, and ensure that the SSL certificates for all the peripheral Servers are installed on the machine where the **hub-xmlrpc-api** service runs. Do this by copying the **RHN-ORG-TRUSTED-SSL-CERT** certificate file from each peripheral Server’ s **http://<server-url>/pub/** directory to **/etc/pki/trust/anchors/**, and run **update-ca-certificates**.

Using the Hub API

Make sure the **hub-xmlrpc-api** service is started:

```
systemctl start hub-xmlrpc-api
```

Once it is running, connect to the service at port 2830 using any XMLRPC-compliant client libraries.

For examples, see **Specialized-guides › Large-deployments**.

Logs are saved in **/var/log/hub/hub-xmlrpc-api.log**. Logs are rotated weekly, or when the log file size reaches the specified limit. By default, the log file size limit is 10 MB.

Hub XMLRPC API Namespaces

The Hub XMLRPC API operates in a similar way to the SUSE Manager API. For SUSE Manager API documentation, see https://documentation.suse.com/suma.

The Hub XMLRPC API exposes the same methods that are available from the server’ s XMLRPC API, with a few differences in parameter and return types. Additionally, the Hub XMLRPC API supports some Hub-specific end points which are not available in the SUSE Manager API.

The Hub XMLRPC API supports three different namespaces:

- The **hub** namespace is used to target the Hub XMLRPC API Server. It supports Hub-specific XMLRPC endpoints which are primarily related to authentication.
- The **unicast** namespace is used to target a single server registered in the hub. It redirects any call transparently to one specific server and returns any value as if the server’ s XMLRPC API endpoint was used directly.
- The **multicast** namespace is used to target multiple peripheral servers registered in the hub. It redirects any call transparently to all the specified servers and returns the results in the form of a **map**.
- If you do not specify a namespace, all calls are transparently redirected to the underlying SUSE Manager Server XMLRPC API of the Hub Server. This allows you to call all available methods on the

SUSE Manager Server XMLRPC API.

Methods called without specifying any of the above namespaces will be forwarded to the normal XMLRPC API of the hub. This is the API exposed on ports 80 and 443.

Some important considerations for hub namespaces:

- Individual server IDs can be obtained using **client.hub.listServerIds(hubSessionKey)**.
- The **unicast** namespace assumes all methods receive **hubSessionKey** and serverID as their first two parameters, then any other parameter as specified by the regular Server API.

  ```
  client.unicast.[namespace].[methodName](hubSessionKey, serverId, param1, param2)
  ```

- The **hubSessionKey** can be obtained using different authentication methods. For more information, see **Specialized-guides › Large-deployments**.
- The **multicast** namespace assumes all methods receive **hubSessionKey**, a list of **ServerID** values, then lists of per-server parameters as specified by the regular server XMLRPC API. The return value is a **map**, with **Successful** and **Failed** entries for each server involved in the call.

  ```
  client.multicast.[namespace].[methodname](hubSessionKey, [serverId1, serverId2],
  [param1_s1, param1_s2], [param2_s1, param2_s2])
  ```

Hub XMLRPC API Authentication Modes

The Hub XMLRPC API supports three different authentication modes:

- Manual mode (default): API credentials must be explicitly provided for each server.
- Relay mode: the credentials used to authenticate with the Hub are also used to authenticate to each server. You must provide a list of servers to connect to.
- Auto-connect mode: credentials are reused for each server, and any peripheral server you have access to is automatically connected.

Authentication Examples

This section provides examples of each authentication method.

Example: Manual Authentication

In manual mode, credentials have to be explicitly provided for each peripheral server before you can connect to it.

A typical workflow for manual authentication is:

1. Credentials for the Hub are passed to the **login** method, and a session key for the Hub is returned (**hubSessionKey**).
2. Using the session key from the previous step, SUSE Manager Server IDs are obtained for all the peripheral servers attached to the Hub via the **hub.listServerIds** method.

3. Credentials for each peripheral server are provided to the **attachToServers** method. This performs authentication against each server’ s XMLRPC API endpoint.
4. A **multicast** call is performed on a set of servers. This is defined by **serverIds**, which contains the IDs of the servers to target. In the background, **system.list_system** is called on each server’ s XMLRPC API
5. Hub aggregates the results and returns the response in the form of a **map**. The map has two entries:
   - **Successful**: list of responses for those peripheral servers where the call succeeded.
   - **Failed**: list of responses for those peripheral servers where the call failed.

If you want to call a method on just one SUSE Manager Server, then Hub API also provides a **unicast** namespace. In this case, the response will be a single value and not a map, in the same way as if you called that SUSE Manager server’ s API directly.

列表 4. Example Python Script for Manual Authentication:

```
#!/usr/bin/python3
import xmlrpc.client

HUB_XMLRPC_API_URL = "<HUB_XMLRPC_API_URL>"
HUB_USERNAME = "<USERNAME>"
HUB_PASSWORD = "<PASSWORD>"

client = xmlrpc.client.ServerProxy(HUB_XMLRPC_API_URL, verbose=0)

hubSessionKey = client.hub.login(HUB_USERNAME, HUB_PASSWORD)

# Get the server IDs
serverIds = client.hub.listServerIds(hubSessionKey)

# For simplicity, this example assumes you are using the same username and password here,
as on the hub server.
# However, in most cases, every server has its own individual credentials.
usernames = [HUB_USERNAME for s in serverIds]
passwords = [HUB_PASSWORD for s in serverIds]

# Each server uses the credentials set above, client.hub.attachToServers needs
# them passed as lists with as many elements as there are servers.
client.hub.attachToServers(hubSessionKey, serverIds, usernames, passwords)

# Perform the operation
systemsPerServer = client.multicast.system.list_systems(hubSessionKey, serverIds)
successfulResponses = systemsPerServer["Successful"]["Responses"]
failedResponses = systemsPerServer["Failed"]["Responses"]

for system in successfulResponses:
  print(system)

# Log out
client.hub.logout(hubSessionKey)
```

Example: Relay Authentication

In relay authentication mode, the credentials used to sign in to the Hub API are also used to sign in into the APIs of the peripheral servers the user wants to work with. In this authentication mode, it is assumed that the same credentials are valid for every server, and that they correspond to a user with appropriate permissions.

After signing in, you must call the **attachToServers** method. This method defines the servers to target in all subsequent calls.

A typical workflow for relay authentication is:

1. Credentials for the Hub are passed to the **loginWithAuthRelayMode** method, and a session key for the Hub is returned (**hubSessionKey**).
2. Using the session key from the previous step, SUSE Manager Server IDs are obtained for all the peripheral servers attached to the Hub via the **hub.listServerIds** method
3. A call to **attachToServers** is made, and the same credentials used to sign in to the Hub are passed to each server. This performs authentication against each server’ s XMLRPC API endpoint.
4. A **multicast** call is performed on a set of servers. This is defined by **serverIds**, which contains the IDs of the servers to target. In the background, **system.list_system** is called on each server’ s XMLRPC API.
5. Hub aggregates the results and returns the response in the form of a **map**. The map has two entries:
    - **Successful**: list of responses for those peripheral servers where the call succeeded.
    - **Failed**: list of responses for those peripheral servers the call failed.

列表 5. Example Python Script for Relay Authentication:

```
#!/usr/bin/python3
import xmlrpc.client

HUB_XMLRPC_API_URL = "<HUB_XMLRPC_API_URL>"
HUB_USERNAME = "<USERNAME>"
HUB_PASSWORD = "<PASSWORD>"

client = xmlrpc.client.ServerProxy(HUB_XMLRPC_API_URL, verbose=0)

hubSessionKey = client.hub.loginWithAuthRelayMode(HUB_USERNAME, HUB_PASSWORD)

# Get the server IDs
serverIds = client.hub.listServerIds(hubSessionKey)

# Authenticate those servers(same credentials will be used as of hub to authenticate)
client.hub.attachToServers(hubSessionKey, serverIds)

# Perform the needed operation
systemsPerServer = client.multicast.system.list_systems(hubSessionKey, serverIds)
successfulResponses = systemsPerServer["Successful"]["Responses"]
failedResponses = systemsPerServer["Failed"]["Responses"]

for system in successfulResponses:
  print(system)

# Log out
client.hub.logout(hubSessionKey)
```

Example: Auto-Connect Authentication

Auto-connect mode is similar to relay mode, it uses the Hub credentials to sign in in to all peripheral servers. However, there is no need to use the **attachToServers** method, as auto-connect mode connects to all available peripheral servers. This occurs at the same time as you sign in to the Hub.

A typical workflow for auto-connect authentication is:

1. Credentials for the Hub are passed to the **loginWithAutoconnectMode** method, and a session key for the Hub is returned (**hubSessionKey**).
2. A **multicast** call is performed on a set of servers. This is defined by **serverIds**, which contains the IDs of the servers to target. In the background, **system.list_system** is called on each server’ s XMLRPC API.
3. Hub aggregates the results and returns the response in the form of a **map**. The map has two entries:
   - **Successful**: list of responses for those peripheral servers where the call succeeded.
   - **Failed**: list of responses for those peripheral servers where the call failed.

列表 6. Example Python Script for Auto-Connect Authentication:

```
#!/usr/bin/python3
import xmlrpc.client

HUB_XMLRPC_API_URL = "<HUB_XMLRPC_API_URL>"
HUB_USERNAME = "<USERNAME>"
HUB_PASSWORD = "<PASSWORD>"

client = xmlrpc.client.ServerProxy(HUB_XMLRPC_API_URL, verbose=0)

loginResponse = client.hub.loginWithAutoconnectMode(HUB_USERNAME, HUB_PASSWORD)
hubSessionKey = loginResponse["SessionKey"]

# Get the server IDs
serverIds = client.hub.listServerIds(hubSessionKey)

# Perform the needed operation
systemsPerServer = client.multicast.system.list_systems(hubSessionKey, serverIds)
successfulResponses = systemsPerServer["Successful"]["Responses"]
failedResponses = systemsPerServer["Failed"]["Responses"]

for system in successfulResponses:
  print(system)

# Log out
client.hub.logout(hubSessionKey)
```

Hub Reporting

The Hub prepares and provides content for multiple peripheral SUSE Manager Servers. The goal of the reporting feature is to get data from these Servers back and have combined reporting data available on the Hub. The data is made available for external Reporting Tools.

Architecture

The main database is a PostgresDB in the SUSE Manager Hub system. It stores all the information collected from all the servers, and aggregates them. Every peripheral Server has its own reporting database where the information is collected for that system. In summary:

- the DB in SUSE Manager Hub stores, collects and eventually aggregates data coming from all the DBs of the peripheral Servers,
- the DB in SUSE Manager Hub stores also its own data from the systems directly connected and managed by the Hub,
- the DB in peripheral SUSE Manager Server stores its own data,
- the reporting tool can be connected either to the Hub or to any SUSE Manager Server.

Setup

The reporting database and schema are set up by default using the local PostgreSQL server. The reporting database is a separate database accessible via the network.

As a requirement we expect all server certificates are signed by the same Root Certificate Authority (CA). The whole SUSE Manager Hub environment should be using only one Root CA.

Create a DB user for the reporting

Before connecting an external Reporting Tools to the Database, a user with read-only permission should be created. For doing that, it is possible to use **uyuni-setup-reportdb-user**.

```
usage: uyuni-setup-reportdb-user [options]

options:
  --help
            show this help message and exit
  --non-interactive
            Switches to non-interactive mode
  --dbuser=DBUSER
            Report DB User
  --dbpassword=DBPASSWORD
            Report DB Password
  --add
            Add the new user
  --delete
            Delete the user
  --modify
            Set a new password
```

Database Schema

The schema exports the most important tables from the main SUSE Manager Database as a de-normalized variant containing only data which are relevant for a report.

Ready-to-use reports are provided as views, aggregating data over multiple tables.

Every table gets an extra id column (**mgm_id**) specifying the SUSE Manager server which provided the data. On a single SUSE Manager Server this column has the standard value **1** which represent **localhost**. On the SUSE Manager Hub it is replaced with the real server id the managed server has in the hub database.

Another common additional field is **synced_date**, which represents the time when the data were exported from the main SUSE Manager Server database.

Schema documentation can be found on the Left Navigation Bar → **Help › Report Database Schema**.

Data Generation and Update

SUSE Manager uses taskomatic jobs to generate the data for the reporting database and to get the data from the peripheral servers to the hub.

To generate and update the data on a peripheral server, the responsible schedule is called **update-reporting-default**. It is executed by default daily at midnight.

On the SUSE Manager Hub there is a second schedule which is important. To fetch the reporting data from all registered peripheral servers, the task with the name **update-reporting-hub-default** is executed daily at 1:30 AM.

All times are in the local timezone of the server. If the peripheral servers are in different timezones, it is recommended to align all the schedules. Make sure that all reporting data are gathered at a specific point in time, and **update-reporting-hub-default** is running when all peripheral servers have actually finished their local jobs.

Tuning Reporting Jobs

report_db_batch_size

Description	The maximum number of rows fetched and written from one database to the other in one shot.
Tune when	Out of memory errors or on processing speed problems.
Value default	2000
Value recommendation	500 - 5000
Location	**/etc/rhn/rhn.conf**
Example	**report_db_batch_size = 4000**
After changing	Check memory usage. Monitor memory usage closely before and after the change.

report_db_hub_workers

Description	The maximum number of workers requesting data from peripheral servers on a hub at the same point in time.
Tune when	The number of peripheral servers increases significantly (more than 100), or a faster synchronization is required.

Value default	2
Value recommendation	2 - 10
Location	**/etc/rhn/rhn.conf**
Example	**report_db_hub_workers = 5**
After changing	Check memory usage. Monitor memory and cpu usage of the hub closely before and after the change. Also monitor the load, cpu and memory usage of the reporting database of the hub.
Notes	All the data collected from the peripheral server must be written into the hub reporting database. Tuning that database could increase the performance as well.

Managing Large Scale Deployments in a Retail Environment

SUSE Manager for Retail 4.3 is an open source infrastructure management solution, optimized and tailored specifically for the retail industry. It uses the same technology as SUSE Manager, but is customized to address the needs of retail organizations.

SUSE Manager for Retail is designed for use in retail situations where customers can use point-of-service terminals to purchase or exchange goods, take part in promotions, or collect loyalty points. In addition to retail installations, it can also be used for novel purposes, such as maintaining student computers in an educational environment, or self-service kiosks in banks or hospitals.

SUSE Manager for Retail is intended for use in installations that include servers, workstations, point-of-service terminals, and other devices. It allows administrators to install, configure, and update the software on their servers, and manage the deployment and provisioning of point-of-service machines.

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. SUSE Manager 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.

For more information about setting up and using SUSE Manager for Retail, see **Retail › Retail-overview**.

Tuning Large Scale Deployments

SUSE Manager is designed by default to work on small and medium scale installations. For installations with more than 1000 clients per SUSE Manager Server, adequate hardware sizing and parameter tuning must be performed.

The instructions in this section can have severe and catastrophic performance impacts when improperly used. In some cases, they can cause SUSE Manager to completely cease functioning. Always test changes before implementing them in a production environment. During implementation, take care when changing parameters. Monitor performance before and after each change, and revert any steps that do not produce the expected result.

We strongly recommend that you contact SUSE Consulting for assistance with tuning.

SUSE will not provide support for catastrophic failure when these advanced parameters are modified without consultation.

Tuning is not required on installations of fewer than 1000 clients. Do not perform these instructions on small or medium scale installations.

The Tuning Process

Any SUSE Manager installation is subject to a number of design and infrastructure constraints that, for the purposes of tuning, we call environmental variables. Environmental variables can include the total number of clients, the number of different operating systems under management, and the number of software channels.

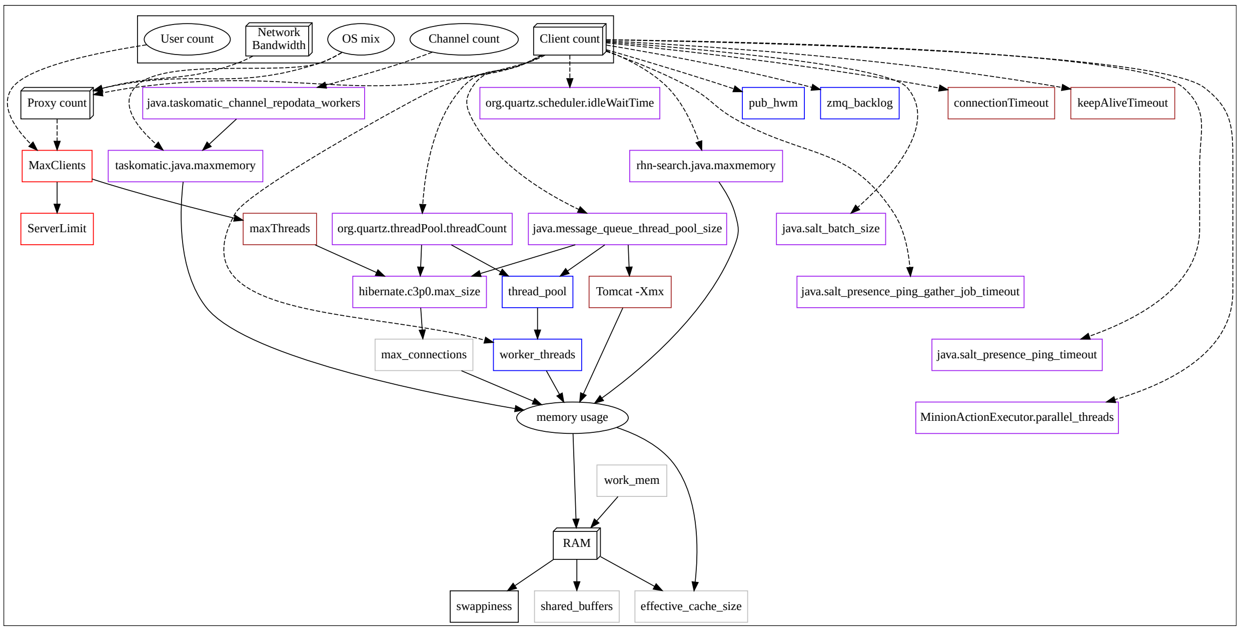
Environmental variables influence, either directly or indirectly, the value of most configuration parameters. During the tuning process, the configuration parameters are manipulated to improve system performance.

Before you begin tuning, you will need to estimate the best setting for each environment variable, and adjust the configuration parameters to suit.

To help you with the estimation process, we have provided you with a dependency graph. Locate the environmental variables on the dependency graph to determine how they will influence other variables and parameters.

Environmental variables are represented by graph nodes in a rectangle at the top of the dependency graph. Each node is connected to the relevant parameters that might need tuning. Consult the relevant sections in this document for more information about recommended values.

Tuning one parameter might require tuning other parameters, or changing hardware, or the infrastructure. When you change a parameter, follow the arrows from that node on the graph to determine what other parameters might need adjustment. Continue through each parameter until you have visited all nodes on the graph.



Key to the Dependency Graph

- 3D boxes are hardware design variables or constraints
- Oval-shaped boxes are software or system design variables or constraints
- Rectangle-shaped boxes are configurable parameters, color-coded by configuration file:
  - Red: Apache **httpd** configuration files
  - Blue: Salt configuration files
  - Brown: Tomcat configuration files
  - Grey: PostgreSQL configuration files
  - Purple: **/etc/rhn/rhn.conf**
- Dashed connecting lines indicate a variable or constraint that might require a change to another parameter
- Solid connecting lines indicate that changing a configuration parameter requires checking another one to prevent issues

After the initial tuning has been completed, you will need to consider tuning again in these cases:

- If your tuning inputs change significantly
- If special conditions arise that require a certain parameter to be changed. For example, if specific warnings appear in a log file.
- If performance is not satisfactory

To re-tune your installation, you will need to use the dependency graph again. Start from the node where significant change has happened.

Environmental Variables

This section contains information about environmental variables (inputs to the tuning process).

Network Bandwidth

A measure of the typically available egress bandwith from the SUSE Manager Server host to the clients or SUSE Manager Proxy hosts. This should take into account network hardware and topology as well as possible capacity limits on switches, routers, and other network equipment between the server and clients.

Channel count

The number of expected channels to manage. Includes any vendor-provided, third-party, and cloned or staged channels.

Client count

The total number of actual or expected clients. It is important to tune any parameters in advance of a client count increase, whenever possible.

OS mix

The number of distinct operating system versions that managed clients have installed. This is ordered by family (SUSE Linux Enterprise, openSUSE, Red Hat Enterprise Linux, or Ubuntu based). Storage and computing requirements are different in each case.

User count

The expected maximum amount of concurrent users interacting with the Web UI plus the number of programs simultaneously using the XMLRPC API. Includes **spacecmd**, **spacewalk-clone-by-date**, and similar.

Parameters

This section contains information about the available parameters.

MaxClients

Description	The maximum number of HTTP requests served simultaneously by Apache httpd. Proxies, Web UI, and XMLRPC API clients each consume one. Requests exceeding the parameter will be queued and might result in timeouts.
Tune when	User count and proxy count increase significantly and this line appears in **/var/log/apache2/error_log**: [⋯] **[mpm_prefork:error] [pid ⋯] AH00161: server reached MaxRequestWorkers setting, consider raising the MaxRequestWorkers setting**.
Value default	150
Value recommendation	150-500

Location	**/etc/apache2/server-tuning.conf**, in the **prefork.c** section
Example	**MaxClients = 200**
After changing	Immediately change **ServerLimit** and check **maxThreads** for possible adjustment.
Notes	This parameter was renamed to **MaxRequestWorkers**, both names are valid.
More information	https://httpd.apache.org/docs/2.4/en/mod/mpm_common.html#maxrequestworkers

ServerLimit

Description	The number of Apache httpd processes serving HTTP requests simultaneously. The number must equal **MaxClients**.
Tune when	**MaxClients** changes
Value default	150
Value recommendation	The same value as **MaxClients**
Location	**/etc/apache2/server-tuning.conf**, in the **prefork.c** section
Example	**ServerLimit = 200**
More information	https://httpd.apache.org/docs/2.4/en/mod/mpm_common.html#serverlimit

maxThreads

Description	The number of Tomcat threads dedicated to serving HTTP requests
Tune when	**MaxClients** changes. **maxThreads** must always be equal or greater than **MaxClients**
Value default	150
Value recommendation	The same value as **MaxClients**
Location	**/etc/tomcat/server.xml**
Example	**<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8" address="127.0.0.1" maxThreads="200" connectionTimeout="20000"/>**

More information	https://tomcat.apache.org/tomcat-9.0-doc/config/http.html

connectionTimeout

Description	The number of milliseconds before a non-responding AJP connection is forcibly closed.
Tune when	Client count increases significantly and **AH00992**, **AH00877**, and **AH01030** errors appear in Apache error logs during a load peak.
Value default	900000
Value recommendation	20000-3600000
Location	**/etc/tomcat/server.xml**
Example	**<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8" address="127.0.0.1" maxThreads="200" connectionTimeout="1000000" keepAliveTimeout="300000"/>**
More information	https://tomcat.apache.org/tomcat-9.0-doc/config/http.html

keepAliveTimeout

Description	The number of milliseconds without data exchange from the JVM before a non-responding AJP connection is forcibly closed.
Tune when	Client count increases significantly and **AH00992**, **AH00877**, and **AH01030** errors appear in Apache error logs during a load peak.
Value default	300000
Value recommendation	20000-600000
Location	**/etc/tomcat/server.xml**
Example	**<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8" address="127.0.0.1" maxThreads="200" connectionTimeout="1000000" keepAliveTimeout="400000"/>**
More information	https://tomcat.apache.org/tomcat-9.0-doc/config/http.html

Tomcat’s -Xmx

Description	The maximum amount of memory Tomcat can use
Tune when	**java.message_queue_thread_pool_size** is increased or **OutOfMemoryException** errors appear in **/var/log/rhn/rhn_web_ui.log**
Value default	1 GiB
Value recommendation	4-8 GiB
Location	**/etc/tomcat/conf.d/tomcat_java_opts.conf**
Example	**JAVA_OPTS="… -Xmx8G …"**
After changing	Check memory usage
More information	https://docs.oracle.com/javase/8/docs/technotes/tools/windows/java.html

java.disable_list_update_status

Description	Disable displaying the update status for clients of a system group
Tune when	displaying the update status causes timeouts
Value default	**false**
Value recommendation	
Location	**/etc/rhn/rhn.conf**
Example	**java.disable_list_update_status = true**
After changing	?
Notes	
More information	**man rhn.conf**

java.message_queue_thread_pool_size

Description	The maximum number of threads in Tomcat dedicated to asynchronous operations
Tune when	Client count increases significantly
Value default	5
Value recommendation	50 - 150
Location	**/etc/rhn/rhn.conf**
Example	**java.message_queue_thread_pool_size = 50**

After changing	Check **hibernate.c3p0.max_size**, as each thread consumes a PostgreSQL connection, starvation might happen if the allocated connection pool is insufficient. Check **thread_pool**, as each thread might perform Salt API calls, starvation might happen if the allocated Salt thread pool is insufficient. Check Tomcat’ s **-Xmx**, as each thread consumes memory, **OutOfMemoryException** might be raised if insufficient.
Notes	Incoming Salt events are handled in separate thread pool, see **java.salt_event_thread_pool_size**
More information	**man rhn.conf**

java.salt_batch_size

Description	The maximum number of minions concurrently executing a scheduled action.
Tune when	Client count reaches several thousands and actions are not executed quickly enough.
Value default	200
Value recommendation	200-500
Location	**/etc/rhn/rhn.conf**
Example	**java.salt_batch_size = 300**
After changing	Check memory usage. Monitor memory usage closely before and after the change.
More information	**Specialized-guides › Salt**

java.salt_event_thread_pool_size

Description	The maximum number of threads in Tomcat dedicated to handling of incoming Salt events.
Tune when	The number of queued Salt events grows. Typically, this can happen during onboarding of large number of minions with higher value of **java.salt_presence_ping_timeout**. The number of events can be queried by **echo "select count(*) from susesaltevent;" | spacewalk-sql --select -mode-direct -**
Value default	8

Value recommendation	20-100
Location	**/etc/rhn/rhn.conf**
Example	**java.salt_event_thread_pool_size = 50**
After changing	Check the length of Salt event queue. Check **hibernate.c3p0.max_size**, as each thread consumes a PostgreSQL connection, starvation might happen if the allocated connection pool is insufficient. Check **thread_pool**, as each thread might perform Salt API calls, starvation might happen if the allocated Salt thread pool is insufficient. Check Tomcat’s **-Xmx**, as each thread consumes memory, **OutOfMemoryException** might be raised if insufficient.
More information	**man rhn.conf**

java.salt_presence_ping_timeout

Description	Before any action is executed on a client, a presence ping is executed to make sure the client is reachable. This parameter sets the amount of time before a second command (in most cases **state.apply** or any other Salt function) is sent to the client to verify its presence. Having many clients typically means some will respond faster than others, so this timeout could be raised to accommodate for the slower ones.
Tune when	Client count increases significantly, or some clients are responding correctly but too slowly, and SUSE Manager excludes them from calls. This line appears in **/var/log/rhn/rhn_web_ui.log**: **"Got no result for <COMMAND> on minion <MINION_ID> (minion did not respond in time)"**
Value default	4 seconds
Value recommendation	4-20 seconds
Location	**/etc/rhn/rhn.conf**
Example	**java.salt_presence_ping_timeout = 10**
After changing	Large **java.salt_presence_ping_timeout** value can reduce overall throughput. This can be compensated by increasing java.salt_event_thread_pool_size
More information	**Specialized-guides › Salt**

java.salt_presence_ping_gather_job_timeout

Description	Before any action is executed on a client, a presence ping is executed to make sure the client is reachable. After **java.salt_presence_ping_timeout** seconds have elapsed without a response, a second command (in most cases **state.apply** or any other Salt function) is sent to the client and if there is no response from the client for the amount of seconds specified with this parameter one more call (**saltutil.find_job**) is sent for a final check. This parameter sets the number of seconds after the second command after which the client is definitely considered timeout. Having many clients typically means some will respond faster than others, so this timeout could be raised to accommodate for the slower ones.
Tune when	Client count increases significantly, or some clients are responding correctly but too slowly, and SUSE Manager excludes them from calls. This line appears in **/var/log/rhn/rhn_web_ui.log**: **"Got no result for <COMMAND> on minion <MINION_ID> (minion did not respond in time)"**
Value default	1 second
Value recommendation	1-50 seconds
Location	**/etc/rhn/rhn.conf**
Example	**java.salt_presence_ping_gather_job_timeout = 20**
More information	**Specialized-guides › Salt**

java.taskomatic_channel_repodata_workers

Description	Whenever content is changed in a software channel, its metadata needs to be recomputed before clients can use it. Channel-altering operations include the addition of a patch, the removal of a package or a repository synchronization run. This parameter specifies the maximum number of Taskomatic threads that SUSE Manager will use to recompute the channel metadata. Channel metadata computation is both CPU-bound and memory-heavy, so raising this parameter and operating on many channels simultaneously could cause Taskomatic to consume significant resources, but channels will be available to clients sooner.
Tune when	Channel count becomes larger than 50, or more concurrent operations on channels are expected.
Value default	2
Value recommendation	2-10
Location	**/etc/rhn/rhn.conf**
Example	**java.taskomatic_channel_repodata_workers = 4**
After changing	Check **taskomatic.java.maxmemory** for adjustment, as every new thread will consume memory
More information	**man rhn.conf**

taskomatic.java.maxmemory

Description	The maximum amount of memory Taskomatic can use. Generation of metadata, especially for some OSs, can be memory-intensive, so this parameter might need raising depending on the managed OS mix.
Tune when	**java.taskomatic_channel_repodata_workers** increases, OSs are added to SUSE Manager (particularly Red Hat Enterprise Linux or Ubuntu), or **OutOfMemoryException** errors appear in **/var/log/rhn/rhn_taskomatic_daemon.log**.
Value default	4096 MiB
Value recommendation	4096-16384 MiB
Location	**/etc/rhn/rhn.conf**

Example	**taskomatic.java.maxmemory = 8192**
After changing	Check memory usage.
More information	**man rhn.conf**

org.quartz.threadPool.threadCount

Description	The number of Taskomatic worker threads. Increasing this value allows Taskomatic to serve more clients in parallel.
Tune when	Client count increases significantly
Value default	20
Value recommendation	20-200
Location	**/etc/rhn/rhn.conf**
Example	**org.quartz.threadPool.threadCount = 100**
After changing	Check **hibernate.c3p0.max_size** and **thread_pool** for adjustment
More information	http://www.quartz-scheduler.org/documentation/2.4.0-SNAPSHOT/configuration.html

org.quartz.scheduler.idleWaitTime

Description	Cycle time for Taskomatic. Decreasing this value lowers the latency of Taskomatic.
Tune when	Client count is in the thousands.
Value default	5000 ms
Value recommendation	1000-5000 ms
Location	**/etc/rhn/rhn.conf**
Example	**org.quartz.scheduler.idleWaitTime = 1000**
More information	http://www.quartz-scheduler.org/documentation/2.4.0-SNAPSHOT/configuration.html

MinionActionExecutor.parallel_threads

Description	Number of Taskomatic threads dedicated to sending commands to Salt clients as a result of actions being executed.
Tune when	Client count is in the thousands.
Value default	1
Value recommendation	1-10
Location	**/etc/rhn/rhn.conf**
Example	**taskomatic.minion_action_executor.parallel_threads = 10**

SSHMinionActionExecutor.parallel_threads

Description	Number of Taskomatic threads dedicated to sending commands to Salt SSH clients as a result of actions being executed.
Tune when	Client count is in the hundreds.
Value default	20
Value recommendation	20-100
Location	**/etc/rhn/rhn.conf**
Example	**taskomatic.sshminion_action_executor.parallel_threads = 40**

hibernate.c3p0.max_size

Description	Maximum number of PostgreSQL connections simultaneously available to both Tomcat and Taskomatic. If any of those components requires more concurrent connections, their requests will be queued.
Tune when	**java.message_queue_thread_pool_size** or **maxThreads** increase significantly, or when **org.quartz.threadPool.threadCount** has changed significantly. Each thread consumes one connection in Taskomatic and Tomcat, having more threads than connections might result in starving.
Value default	20

Value recommendation	100 to 200, higher than the maximum of **java.message_queue_thread_pool_size** + **maxThreads** and **org.quartz.threadPool.threadCount**
Location	**/etc/rhn/rhn.conf**
Example	**hibernate.c3p0.max_size = 100**
After changing	Check **max_connections** for adjustment.
More information	https://www.mchange.com/projects/c3p0/#maxPoolSize

rhn-search.java.maxmemory

Description	The maximum amount of memory that the **rhn-search** service can use.
Tune when	Client count increases significantly, and **OutOfMemoryException** errors appear in **journalctl -u rhn-search**.
Value default	512 MiB
Value recommendation	512-4096 MiB
Location	**/etc/rhn/rhn.conf**
Example	**rhn-search.java.maxmemory = 4096**
After changing	Check memory usage.

shared_buffers

Description	The amount of memory reserved for PostgreSQL shared buffers, which contain caches of database tables and index data.
Tune when	RAM changes
Value default	25% of total RAM
Value recommendation	25-40% of total RAM
Location	**/var/lib/pgsql/data/postgresql.conf**
Example	**shared_buffers = 8192MB**
After changing	Check memory usage.
More information	https://www.postgresql.org/docs/15/runtime-config-resource.html#GUC-SHARED-BUFFERS

max_connections

Description	Maximum number of PostgreSQL connections available to applications. More connections allow for more concurrent threads/workers in various components (in particular Tomcat and Taskomatic), which generally improves performance. However, each connection consumes resources, in particular **work_mem** megabytes per sort operation per connection.
Tune when	**hibernate.c3p0.max_size** changes significantly, as that parameter determines the maximum number of connections available to Tomcat and Taskomatic
Value default	400
Value recommendation	Depends on other settings, use **/usr/lib/susemanager/bin/susemanager-connection-check** to obtain a recommendation.
Location	**/var/lib/pgsql/data/postgresql.conf**
Example	**max_connections = 250**
After changing	Check memory usage. Monitor memory usage closely before and after the change.
More information	https://www.postgresql.org/docs/15/runtime-config-connection.html#GUC-MAX-CONNECTIONS

work_mem

Description	The amount of memory allocated by PostgreSQL every time a connection needs to do a sort or hash operation. Every connection (as specified by **max_connections**) might make use of an amount of memory equal to a multiple of **work_mem**.
Tune when	Database operations are slow because of excessive temporary file disk I/O. To test if that is happening, add **log_temp_files = 5120** to **/var/lib/pgsql/data/postgresql.conf**, restart PostgreSQL, and monitor the PostgreSQL log files. If you see lines containing **LOG: temporary file:** try raising this parameter’ s value to help reduce disk I/O and speed up database operations.
Value recommendation	2-20 MB

Location	**/var/lib/pgsql/data/postgresql.conf**
Example	**work_mem = 10MB**
After changing	check if the SUSE Manager Server might need additional RAM.
More information	https://www.postgresql.org/docs/15/runtime-config-resource.html#GUC-WORK-MEM

effective_cache_size

Description	Estimation of the total memory available to PostgreSQL for caching. It is the explicitly reserved memory (**shared_buffers**) plus any memory used by the kernel as cache/buffer.
Tune when	Hardware RAM or memory usage increase significantly
Value recommendation	Start with 75% of total RAM. For finer settings, use **shared_buffers** + free memory + buffer/cache memory. Free and buffer/cache can be determined via the **free -m** command (**free** and **buff/cache** in the output respectively)
Location	**/var/lib/pgsql/data/postgresql.conf**
Example	**effective_cache_size = 24GB**
After changing	Check memory usage
Notes	This is an estimation for the query planner, not an allocation.
More information	https://www.postgresql.org/docs/15/runtime-config-query.html#GUC-EFFECTIVE-CACHE-SIZE

thread_pool

Description	The number of worker threads serving Salt API HTTP requests. A higher number can improve parallelism of SUSE Manager Server-initiated Salt operations, but will consume more memory.
Tune when	**java.message_queue_thread_pool_size** or **org.quartz.threadPool.threadCount** are changed. Starvation can occur when there are more Tomcat or Taskomatic threads making simultaneous Salt API calls than there are Salt API worker threads.

Value default	100
Value recommendation	100-500, but should be higher than the sum of **java.message_queue_thread_pool_size** and **org.quartz.threadPool.threadCount**
Location	**/etc/salt/master.d/susemanager.conf**, in the **rest_cherrypy** section.
Example	**thread_pool: 100**
After changing	Check **worker_threads** for adjustment.
More information	https://docs.saltproject.io/en/latest/ref/netapi/all/salt.netapi.rest_cherrypy.html#performance-tuning

worker_threads

Description	The number of **salt-master** worker threads that process commands and replies from minions and the Salt API. Increasing this value, assuming sufficient resources are available, allows Salt to process more data in parallel from minions without timing out, but will consume significantly more RAM (typically about 70 MiB per thread). Setting this value to very high values could cause opposite effect as the workers will compete to each other for the CPU resources and the performance could be dropped significantly.
Tune when	Client count increases significantly, **thread_pool** increases significantly, or **SaltReqTimeoutError** or **Message timed out** errors appear in **/var/log/salt/master** could be a sign of too low or too high value of this parameter.
Value default	8
Value recommendation	8-32, depending on the number of the CPU cores available for the server, it is recommended to keep the value slightly less than the number of CPU cores.
Location	**/etc/salt/master.d/tuning.conf**
Example	**worker_threads: 16**

After changing	Check memory usage. Monitor memory usage closely before and after the change. It makes sense to monitor the **salt-master** stats event by enabling **master_stats** and adjusting **master_stats_event_iter** to fine tune the value of this parameter.
More information	https://docs.saltproject.io/en/latest/ref/configuration/master.html#worker-threads

auth_events

Description	Determines whether the master will fire authentication events. Authentication events are fired when a minion performs an authentication check with the master. It helps to reduce the number of events published with the Salt Master Event Publisher and reduce the workload on Event Publisher subscribers.
Tune when	Large amount of **salt/auth** events published in the Salt event bus, which in most cases are useless for the subscribers.
Value default	True
Value recommendation	False
Location	**/etc/salt/master.d/tuning.conf**
Example	**auth_events: False**
More information	https://docs.saltproject.io/en/latest/ref/configuration/master.html#auth-events

minion_data_cache_events

Description	Determines whether the master will fire minion data cache events (**minion/refresh/***). Minion data cache events are fired when a minion requests a minion data cache refresh. It helps to reduce the number of events published with the Salt Master Event Publisher and reduce the workload on Event Publisher subscribers.
Tune when	Large amount of **minion/refresh/*** events published in the Salt event bus, which in most cases are useless for the subscribers.
Value default	True

Value recommendation	False
Location	**/etc/salt/master.d/tuning.conf**
Example	**minion_data_cache_events: False**
More information	https://docs.saltproject.io/en/latest/ref/configuration/master.html#minion-data-cache-events

pub_hwm

Description	The maximum number of outstanding messages sent by **salt-master**. If more than this number of messages need to be sent concurrently, communication with clients slows down, potentially resulting in timeout errors during load peaks.
Tune when	Client count increases significantly and **Salt request timed out. The master is not responding.** errors appear when pinging minions during a load peak.
Value default	1000
Value recommendation	10000-100000
Location	**/etc/salt/master.d/tuning.conf**
Example	**pub_hwm: 10000**
More information	https://docs.saltproject.io/en/latest/ref/configuration/master.html#pub-hwm, https://zeromq.org/socket-api/#high-water-mark

zmq_backlog

Description	The maximum number of allowed client connections that have started but not concluded the opening process. If more than this number of clients connects in a very short time frame, connections are dropped and clients experience a delay re-connecting.
Tune when	Client count increases significantly and very many clients reconnect in a short time frame, TCP connections to the **salt-master** process get dropped by the kernel.

Value default	1000
Value recommendation	1000-5000
Location	**/etc/salt/master.d/tuning.conf**
Example	**zmq_backlog: 2000**
More information	https://docs.saltproject.io/en/latest/ref/configuration/master.html#zmq-backlog, http://api.zeromq.org/3-0:zmq-getsockopt (**ZMQ_BACKLOG**)

swappiness

Description	How aggressively the kernel moves unused data from memory to the swap partition. Setting a lower parameter typically reduces swap usage and results in better performance, especially when RAM memory is abundant.
Tune when	RAM increases, or swap is used when RAM memory is sufficient.
Value default	60
Value recommendation	1-60. For 128 GB of RAM, 10 is expected to give good results.
Location	**/etc/sysctl.conf**
Example	**vm.swappiness = 20**
More information	https://documentation.suse.com/sles/15-SP4/html/SLES-all/cha-tuning-memory.html#cha-tuning-memory-vm

wait_for_backend

Description	Determines whether the **salt-broker** service should wait for backend sockets to be connected before opening the sockets for listening for connections from **salt-minions**. When enabled, it helps to prevent collecting ZeroMQ messages with the internal buffers of the sockets and pushing them to the **salt-master** once connection is restored.
Tune when	Unstable connectivity between the SUSE Manager Proxy and the SUSE Manager Server.
Value default	False

Value recommendation	True
Location	**/etc/salt/broker**
Example	**wait_for_backend: True**
More information	Proxies Connectivity

tcp_keepalive

Description	The tcp keepalive interval to set on TCP ports. This setting can be used to tune Salt connectivity issues in messy network environments with misbehaving firewalls.
Tune when	Unstable connectivity between managed clients and the SUSE Manager Proxy or the SUSE Manager Server.
Value default	True
Value recommendation	True
Location	**/etc/venv-salt-minon/minion.d/tuning.conf** or **/etc/salt/minion.d/tuning.conf**, depending on the minion type.
Example	**tcp_keepalive: True**
After changing	Check Minions Connectivity for more details to fine tune extra keepalive parameters.
More information	https://docs.saltproject.io/en/latest/ref/configuration/minion.html#tcp-keepalive, Minions Connectivity

Memory Usage

Adjusting some of the parameters listed in this section can result in a higher amount of RAM being used by various components. It is important that the amount of hardware RAM is adequate after any significant change.

To determine how RAM is being used, you will need to check each process that consumes it.

Operating system

Stop all SUSE Manager services and inspect the output of **free -h**.

Java-based components

This includes Taskomatic, Tomcat, and **rhn-search**. These services support a configurable memory cap.

The SUSE Manager Server

Depends on many factors and can only be estimated. Measure PostgreSQL reserved memory by checking **shared_buffers**, permanently. You can also multiply **work_mem** and **max_connections**, and multiply by three for a worst case estimate of per-query RAM. You will also need to check the operating system buffers and caches, which are used by PostgreSQL to host copies of database data. These often automatically occupy any available RAM.

It is important that the SUSE Manager Server has sufficient RAM to accommodate all of these processes, especially OS buffers and caches, to have reasonable PostgreSQL performance. We recommend you keep several gigabytes available at all times, and add more as the database size on disk increases.

Whenever the expected amount of memory available for OS buffers and caches changes, update the **effective_cache_size** parameter to have PostgreSQL use it correctly. You can calculate the total available by finding the total RAM available, less the expected memory usage.

To get a live breakdown of the memory used by services on the SUSE Manager Server, use this command:

```
pidstat -p ALL -r --human 1 60 | tee pidstat-memory.log
```

This command will save a copy of displayed data in the **pidstat-memory.log** file for later analysis.

Proxy

cache_dir

Description	The amount of storage space for the squid cache directory.
Tune when	More than the default storage space is required.
Value default	100
Value recommendation	60 % of available free space
Location	**/etc/squid/squid.conf**
Example	cache_dir aufs /var/cache/squid 15000 16 256
After changing	
More information	[path]``/etc/squid/squid.conf.documented''

In general, SUSE recommends to adjust the value for the cache directory to about 60 % of available free space. Users can set the **cache_dir** option in the **squid.conf** manually.

Monitoring Large Scale Deployments

You can monitor your SUSE Manager environment using Prometheus and Grafana. SUSE Manager Server and Proxy are able to provide self-health metrics. You can also install and manage a number of

Prometheus exporters on Salt clients.

Prometheus and Grafana packages are included in the SUSE Manager Client Tools for SUSE Linux Enterprise 12, SUSE Linux Enterprise 15, Red Hat Enterprise Linux 7, Red Hat Enterprise Linux 8 and openSUSE 15.x.

You need to install Prometheus and Grafana on a machine separate from the SUSE Manager Server. We recommend you use a managed Salt client as your monitoring server.

For more information on monitoring, see **Administration › Monitoring**.

Quick Start: SAP Overview

Updated: 2025-07-21

This guide shows you how to use SUSE Manager to install and configure an SAP cluster. It guides you through setting up a single SUSE Manager Server, preparing your client systems, and configuring the cluster using formulas.

- For more information about SAP, see the SAP documentation at https://documentation.suse.com/sles-sap.
- For more information about SUSE Manager, see the SUSE Manager documentation at https://documentation.suse.com/suma.

Prepare Server

Before you start you need to install the SUSE Manager Server. The method for installing the SUSE Manager Server varies depending on your hardware and environment.

SUSE Manager is installed using the SUSE Linux Enterprise Server 15 Unified Installer. During the installation process, when you are prompted for which product to install, select SUSE Manager Server. The server does not need to have the SUSE Linux Enterprise Server 15 with SAP product installed. For more information about installing the SUSE Manager Server, see **Installation-and-upgrade › Install-server-unified**.

When the SUSE Manager Server is installed, set it up by running the **yast2 susemanager_setup** command from the command prompt. The setup script prompts you to complete additional details about your server, and give you the URL to use to access the Web UI. For more information about setup, see **Installation-and-upgrade › Server-setup**.

You need to do some configuration to set up the SUSE Manager Web UI. In your browser, navigate to the URL of the server, and configure your administration access to the Web UI. For more information about setting up the Web UI, see **Installation-and-upgrade › Webui-setup**.

Now you can use the Web UI to prepare software channels and activation keys for your clients.

On the SUSE Manager Server, add the appropriate SAP channels: From the Web UI, add **SUSE Linux Enterprise Server 15 for SAP**.

Synchronize the SUSE Manager Server with the SUSE Customer Center. You can do this using the Web UI. Add the new channel to your activation key.

To check if a channel has finished synchronizing navigate to **Admin › Setup Wizard** and select the **Products** tab. This dialog displays a completion bar for each product when they are being synchronized.

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

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

Preparing Clients

Your SAP cluster requires several client systems. Prepare your clients on physical or virtual hardware, and ensure you have SUSE Linux Enterprise Server 15 for SAP installation media ready. You cannot create an SAP cluster without the SUSE Linux Enterprise Server SAP extension, as it provides tooling specific to SAP.

One of the key features of SAP is high availability of the cluster. Every component within an SAP cluster has redundancy and failover protection. When you are preparing your clients, ensure you have enough hardware and infrastructure to allow for this. For more information about hardware requirements, see https://documentation.suse.com/sles-sap/15-SP4/html/SLES-SAP-installation/cha-plan.html#sec-hardware

For more information about the clients you need to set up for an SAP cluster, see https://documentation.suse.com/sbp/all.

Register Clients to the SUSE Customer Center

Each client within your SAP cluster must be registered with the SUSE Customer Center. To obtain your registration code, navigate to https://scc.suse.com/login in your web browser. Log in to your SCC account, or follow the prompts to create a new account. Click the **[Subscriptions]** tab to see the registration code. When you install SUSE Linux Enterprise Server 15 for SAP the Unified Installer prompts you for the code.

For more information about registering SUSE Manager with SUSE Customer Center, see **Installation-and-upgrade › General-requirements**.

Configure the Clients for Clustering

Every client system must have all the other client systems listed in their **/etc/hosts** file. Open the **/etc/hosts** file on each client, and add the hostname for each of the other clients.

Create a Shared Storage Device

Each of the clients needs to be able to access a shared disk. The shared disk can be physical hardware connected by ethernet, or you can set up a virtual disk and access it with iSCSI.

If you use a virtual disk, consider hosting it on a separate system. Do not use a client machine to host the shared storage disk.

Download the SAP Installation Software

Download the SAP installation media and save a copy on each client. The software that you require differs depending on your environment. For example, if you are using HANA, you need the SAP HANA platform. If you are using Netweaver, you need different packages. These software packages are provided by SAP, not by SUSE.

Ensure you have saved the installation software in the same file system location on each client. Alternatively, save it to a shared NFS drive.

Configure Clients to Use Latest module.run

Each client needs to be configured to use the latest version of **module.run**. On each of the client machines, open the **/etc/salt/minion** configuration file and add or edit this line:

```
use_superseded:
  - module.run
```

Restart the **salt-minion** process to enable the changes:

```
systemctl restart salt-minion
```

Install Additional Disks for HANA

For the clients that are going to run the HANA database, you require an additional storage device. This device is used to store files required by HANA, which are located in the **/hana/** directory.

We recommend that this storage device be at least 20 GB. For some installations, you might require more, and it is possible to use multiple disks to provide this storage. For comprehensive hardware requirements, see https://documentation.suse.com/sbp/all.

Register Clients to the Server

First of all, make sure you have an activation key that is associated with the **SLE-Product-SLES_SAP15** base channel. For more information about activation keys, see **Client-configuration › Activation-keys**.

In the SUSE Manager Web UI, navigate to **Systems › Bootstrapping**. Fill in the appropriate details, and make sure you check the **Manage System Completely via SSH** checkbox. In the **Activation Key** field, select the SLES for SAP activation key.

For more information about registering, see **Client-configuration › Registration-webui**.

Configure Clients

SUSE Manager uses formulas with forms to configure your SAP clients. There are two formulas that you need to use:

- **Hana** to configure the HANA database
- **Cluster** to configure the clients into a cluster

The formulas are provided by packages that you can download with your package manager. You need to install the formulas on the SUSE Manager Server. When you have installed the package, you can use the SUSE Manager Web UI to enable and configure the formulas. As you go through the formula configuration process, provide details of the clients that contain your SAP cluster, to set them up appropriately.

To install the formulas on the SUSE Manager Server, use your package manager to install these packages:

- **saphanabootstrap-formula**
- **sapnwbootstrap-formula**
- **drbd-formula**
- **habootstrap-formula**
- **salt-shaptools**

The order that you enable and configure the formulas is important. You must enable, configure, and apply the HANA formula first. Then you can enable, configure, and apply the cluster formula. If you perform these steps in the wrong order, your SAP installation fails.

Enable and Configure the HANA Formula

In the SUSE Manager Web UI, navigate to **Systems › System List** and click the client to use as the primary client in the cluster.

Navigate to the **Formulas** tab, locate the **Sap Hana Deployment** heading, and check the **Saphanabootstrap** formula in the list. Click **[Save]** and apply the highstate to activate the formula.

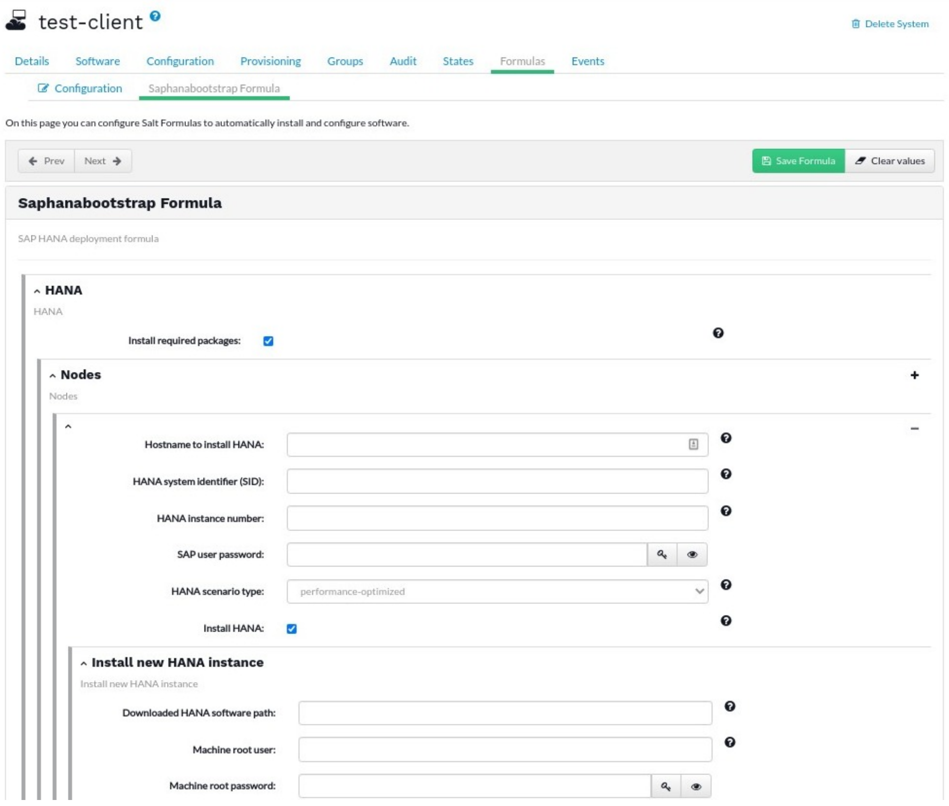
When the formula is activated, navigate to the **Formulas › Hana** tab, and complete the details in the form.

Make sure you check **Install required packages** to install everything you need on the client. In the **Nodes** sections, type the short hostname of the client to install the HANA database or the hostname you can retrieve on the command line with:

```
salt '<client-name>' grains.item host
```

Provide further details for the installation.

Complete the remaining details according to your environment, click **[Save]**, and apply the highstate. When the highstate is complete, you can go on to apply the cluster formula.



test-client

Delete System

On this page you can configure Salt Formulas to automatically install and configure software.

Saphanabootstrap Formula

SAP HANA deployment formula

HANA

HANA

Install required packages: ✓

Nodes

Nodes

Hostname to install HANA:	
HANA system identifier (SID):	
HANA instance number:	
SAP user password:	
HANA scenario type:	performance-optimized
Install HANA:	✓

Install new HANA instance

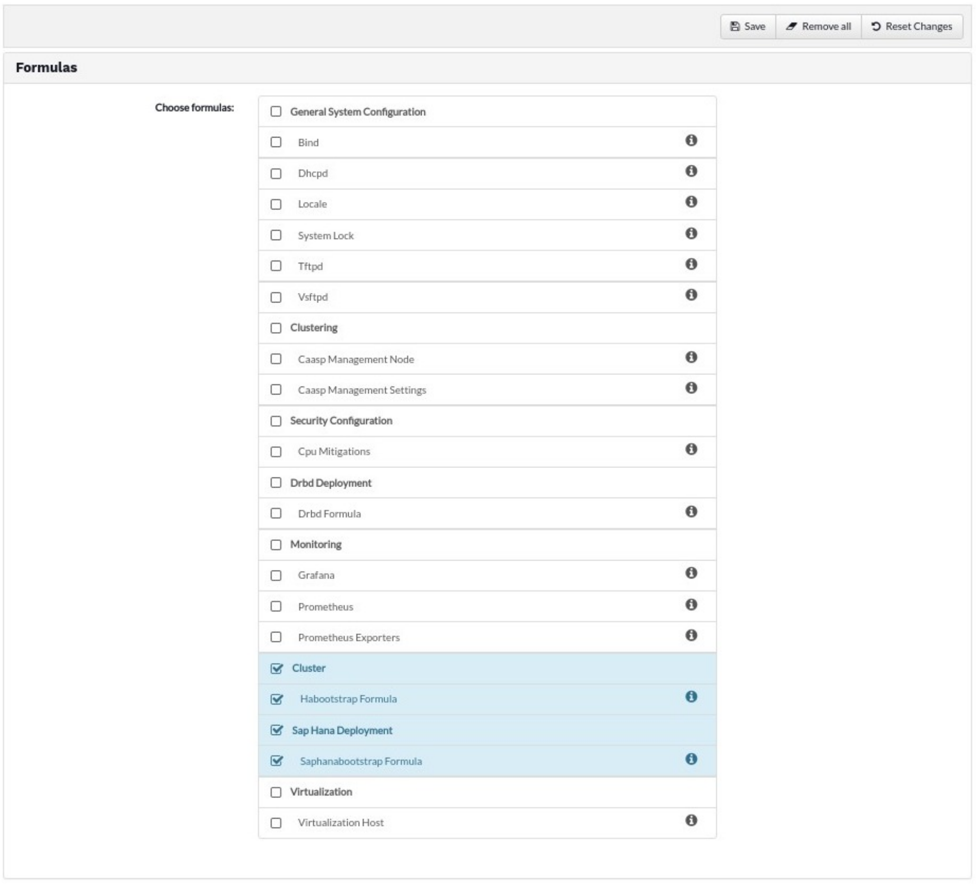
Install new HANA instance

Downloaded HANA software path:	
Machine root user:	
Machine root password:	

Enable and Configure the Cluster Formula

In the SUSE Manager Web UI, navigate to **Systems › System List** and click the client to use as the primary client in the cluster.

Navigate to the **Formulas** tab, locate the **Cluster** heading, and check the **Habootstrap** formula in the list. Click **[Save]** and apply the highstate to activate the formula.


Save | Remove all | Reset Changes

Formulas

Choose formulas:

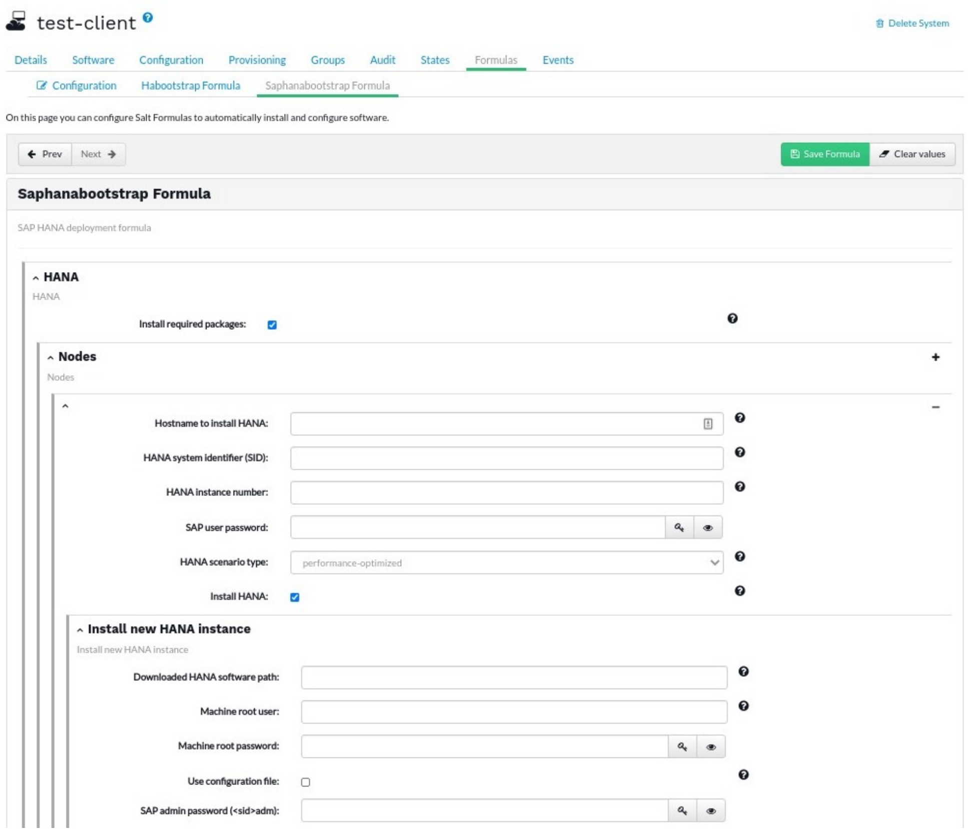
- ☐ General System Configuration
  - ☐ Bind
  - ☐ Dhcpd
  - ☐ Locale
  - ☐ System Lock
  - ☐ Tftpd
  - ☐ Vsftpd
- ☐ Clustering
  - ☐ Caasp Management Node
  - ☐ Caasp Management Settings
- ☐ Security Configuration
  - ☐ Cpu Mitigations
- ☐ Drbd Deployment
  - ☐ Drbd Formula
- ☐ Monitoring
  - ☐ Grafana
  - ☐ Prometheus
  - ☐ Prometheus Exporters
- [✓ Cluster]
  - [✓ Habootstrap Formula]
- [✓ Sap Hana Deployment]
  - [✓ Saphanabootstrap Formula]
- ☐ Virtualization
  - ☐ Virtualization Host

When the formula is activated, navigate to the **Formulas › Cluster** tab, and complete the details in the form.

Make sure you check **Install required packages** to install everything you need on the client. Give your cluster a name, and specify the hostname of the primary client in the cluster.

Complete the remaining details according to your environment, click **[Save]**, and apply the highstate.

Connect the client to itself and forward the standard input/output of the server to the SSH port of the client (-W client:22).



On this page you can configure Salt Formulas to automatically install and configure software.

Saphanabootstrap Formula

SAP HANA deployment formula

HANA

HANA

Install required packages: ☑

Nodes

Nodes

Hostname to install HANA:

HANA system identifier (SID):

HANA instance number:

SAP user password:

HANA scenario type: performance-optimized

Install HANA: ☑

Install new HANA instance

Install new HANA instance

Downloaded HANA software path:

Machine root user:

Machine root password:

Use configuration file: ☐

SAP admin password (<sid>adm):

GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the

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.

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.

ADDENDUM: How to use this License for your documents