Salt Guide

Uyuni 4.0

March 16, 2019

Table of Contents

DRAFT

GNU Free Documentation License

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections

then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
   Permission is granted to copy, distribute and/or modify this document
   under the terms of the GNU Free Documentation License, Version 1.2
   or any later version published by the Free Software Foundation;
   with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
   A copy of the license is included in the section entitled{ldquo}GNU
   Free Documentation License{rdquo}.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “ with…Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
   Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

DRAFT

Introduction

This section introduces you to the Salt features added in Uyuni 3. This chapter assumes you have completed all previous Getting Started sections. At a minimum have the following setup:

- A freshly installed Uyuni server with a main admin account and a synced product channel
- Preferably two registered Salt minions to experiment with.

This guide does not attempt to cover all that Salt has to offer. This guide is a primer for using Salt with Uyuni. For comprehensive Salt documentation, see https://docs.saltstack.com/en/latest/contents.html.

The current version of Salt in Uyuni is 2018.3.0.

DRAFT

Salt Terminology and Commands

Salt Terminology

Grains

Grains provide information about the hardware of a minion. For example, the operating system, IP addresses, network interfaces, memory, etc. When running a Salt command from keep in mind any modules and functions called are run locally from the system being called. Salt modules are stored on minions and master within the following directory:

```
/usr/lib/python2.7/site-packages/salt/
```

List all available grains with the `grains.ls` function:

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

List collected grain system data by using the `grains.ls` function:

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

For more information on grains, see https://docs.saltstack.com/en/latest/topics/grains/.

States

States are templates which place systems into a known configuration, for example which applications and services are installed and running on those systems. States are a way for you to describe what each of your systems should look like. Once written, states are applied to target systems automating the process of managing and maintaining a large numbers of systems into a known state. For more information on states, see https://docs.saltstack.com/en/latest/topics/tutorials/starting_states.html.

Updating Salt

Do not update salt itself using Salt states. First update all other system packages using Salt states then update salt as a separate stand-alone step from the Uyuni Web UI.

Pillar

Pillars unlike grains are created on the master. Pillar files contain information about a minion or group of minions. Pillars allow you to send confidential information to a targeted minion or group of minions. Pillars are useful for sensitive data, configuration of minions, variables, and any arbitrary data which should be defined. For more information on pillars, see https://docs.saltstack.com/en/latest/topics/tutorials/pillar.html.

Beacons

Beacons allow an admin to use the event system in Salt to monitor non-Salt processes. Minions may

use beacons to hook into many types of system proceses for constant monitoring. Once a targeted monitored activity occurs an event is sent on the Salt event bus that may be used to trigger a reactor.

Enabling Beacons

To work with beacons on Salt minions the package python-pyinotify must be installed for SUSE systems. For RES systems install python-inotify. This package is not installed automatically during the salt minion package installation.

Peer Communication with salt-broker

The salt-broker acts like a switch and not like a hub, therefore Peer communication will only work for minions behind the same broker/Proxy. For more information on Salt and peer communication see: https://docs.saltstack.com/en/latest/ref/peer.html

Salt Calls

Salt Calls

Salt calls are defined by three main properties:

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

Target

Use the second property in a Salt call to target a single machine or group of machines. Specify the minion or group of minions you would like to run a function on.

General Targeting

List available grains on all minions:

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

Ping a specific minion:

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

Glob Targeting

Ping all minions using a domain:

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

Display the OS name of all minions with the `webserver` label:

```
salt 'webserver*' grains.item oscodename
```

List Targeting

```
salt -L 'webserver.example.com,db.example.com' test.ping
```

Regular Expression Targeting

You may use PCRE-compliant regular expressions:

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

IP Address Targeting

List minion IP addresses:

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

Ping a specific minion IP address:

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

Ping all minions on a subnet:

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

Lookup a Subnet Using the `ip` Command

You can use the `ip` command to find the subnet mask in the format of `192.168.1.1/24`:

```
ip -o -f inet addr show | awk '/scope global/ {print $4}'
```

Function

Once you have specified a target, provide the function you would like to call. Functions also accept arguments. Arguments are space-delimited, for example:

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

Locating Additional Minion Functions

Find more functions which can be called on minions by running:

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

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

Arguments

Provides the extra data needed by a function you are calling. The command `pkg.install` requires an argument specifying a package to install. YaST has been selected for installation, for example:

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

Salt Commands

The following list provides several useful Salt commands.

salt-run

Print a list of all minions that are up:

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

Print a list of all minions that are down:

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

Print a list with the current status of all Salt minions:

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

Check the version of Salt running on the master and active minions:

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

salt-cp

Copy a file to a minion or set of minions.

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

For more information, see https://docs.saltstack.com/en/latest/ref/cli/salt-cp.html.

salt-key -l

List public keys:

```
salt-key -l
```

salt-key -A

Accept all pending keys:

```
salt-key -A
```

DRAFT

Configuration Management with Salt

Configuration Management Overview

Salt is capable of applying states by matching minions with relevant state data. This data comes from Uyuni in the form of package and custom states.

State Data: Levels of Hierarchy

State data comes from Uyuni in the form of package and custom states and targets minions at three specific levels of hierarchy. The state hierarchy is defined by the following order or priority: individual minions have priority on packages and custom states over groups; next a group has priority over the organization.

- Minion Level

  Systems › Specific Minion › States

- Group Level

  Systems › System Groups

- Organization Level

  Systems › Manage System Types: › My Organization

For example:

- Org1 requires that vim version 1 is installed
- Group1 requires that vim version 2 is installed
- Group2 requires any version installed

This would lead to the following order of hierarchy:

- Minion1 part of [Org1, Group1] wants vim removed, vim is removed (Minion Level)
- Minion2 part of [Org1, Group1] wants vim version 2 gets version 2 (Group Level)
- Minion3 part of [Org1, Group1] wants any version, gets version 2 (Org Level)
- Minion4 part of[Org1, Group2] wants any version, gets vim version 1 (Org Level)

Salt States Storage Locations

The Uyuni salt-master reads its state data from three file root locations.

The directory `/usr/share/susemanager/salt` is used by Uyuni and comes from the susemanager-

sls. It is shipped and updated together with Uyuni and includes certificate setup and common state logic to be applied to packages and channels.

The directory `/srv/susemanager/salt` is generated by Uyuni and based on assigned channels and packages for minions, groups and organizations. This file will be overwritten and regenerated. This could be thought of as the Uyuni database translated into salt directives.

The third directory `/srv/salt` is for custom state data, modules etc. Uyuni does not operate within or utilize this directory. However the state data placed here affects the Highstate of minions and is merged with the total state result generated by Uyuni.

Uyuni States

All sls files created by users will be saved to disk on the salt-master server. These files will be placed in `/srv/susemanager/salt/` and each organization will be placed within its own directory. Although these states are custom, these states are created using Uyuni . The following provides an overview of directory structure:

```
├──── manager_org_DEVEL
│     ├──── files
│     │     ... files needed by states (uploaded by users)...
│     └──── state.sls
          ... other sls files (created by users)...
E.g.:
├──── manager_org_TESTING
│     ├──── files
│     │     └──── motd     # user created
│     │     ... other files needed by states ...
│     └──── motd.sls     # user created
            ... other sls files ...
```

Pillar Data

SUSE Manager exposes a small amount of internal data as Pillars which can be used with custom SUSE Linux Enterprise Server states. Data that is exposed includes group membership, organization membership, and file roots. These are managed either automatically by Uyuni, or manually by the user.

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 minions 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
```

Group States

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

In order 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
```

In these examples we will 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 minions in the group:

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

Salt Formulas

This chapter provides an introduction for using Salt Formulas with Uyuni. Creation of custom formulas will also be introduced.

What are Salt Formulas?

Formulas are collections of Salt States that have been pre-written by other Salt users and contain generic parameter fields. Formulas allow for reliable reproduction of a specific configuration again and again. Formulas can be installed from RPM packages or an external git repository.

This list will help you decide whether to use a state or a formula:

Formula Tips

- When writing states for trivial tasks, formulas are probably not worth the time investment.
- For large, non-trivial configurations use formulas.
- Formulas and States both act as a kind of configuration documentation. Once written and stored you will have a snapshot of what your infrastructure should look like.
- Pre-written formulas are available from the Saltstack formula repository on Github. Use these as a starting point for your own custom formulas.
- Formula data can be managed via the XMLRPC API.

Formula with Forms Improvements

Forms are a graphical representation of the formulas parameter data. You can customize these configuration data in the Uyuni Web UI, with entry fields, drop-down, check boxes, etc.

For more information, see https://www.suse.com/c/forms-formula-success/.

Installing Salt Formulas via RPM

SUSE releases formulas as RPM packages. Available formulas can be located within the `SUSE-Manager-Server-3.2-Pool` channel.

Salt State Name Clashes

If a Salt 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 Salt Formulas from an RPM

1. To search for available formulas, execute the following command on your Uyuni server:

```
zypper se --type package formula
```

You will see a list of available Salt formulas:

```
S | Name              | Summary                                                          |
Type
--+-------------------+------------------------------------------------------------------+
----------
  | locale-formula    | Locale Salt Formula for SUSE Manager                             |
package
```

2. For more information about a formula, run the following command:

```
zypper info locale-formula
```

```
Information for package locale-formula:
---------------------------------------
Repository: SUSE-Manager-Server-{productnumber}-Pool
Name:  locale-formula
Version:  0.2-1.1
Arch: noarch
Vendor:  SUSE LLC <https://www.suse.com/>
Support Level: Level 3
Status: not installed
Installed Size: 47.9 KiB
Installed: No
Source package : locale-formula-0.2-1.1.src
Summary        : Locale Salt Formula for SUSE Manager
Description    :
    Salt Formula for SUSE Manager. Sets up the locale.
```

3. To install a formula run as root:

```
zypper in locale-formula
```

File Structure Overview

RPM-based formulas must be placed in a specific directory structure to ensure proper functionality. A formula always consists of two separate directories: The `states` directory and the `metadata` directory. Folders in these directories need to have an exactly matching name, for example `locale`.

The Formula State Directory

The formula states directory contains anything necessary for a Salt state to work independently. This includes `.sls` files, a `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 locale-formula states directory is located in:

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

The Formula Metadata Directory

The metadata directory contains a `form.yml` file which defines the forms for Uyuni and an optional `metadata.yml` file that can contain additional information about a formula. For example, the locale-formula metadata directory is located in:

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

Custom Formulas

Custom formula data or (non-RPM) formulas need to be placed into any state directory configured as a Salt file root:

State directory

Custom state formula data needs to be placed in:

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

Metadata Directory

Custom metadata (information) needs to be placed in:

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

All custom folders located in the following directories need to contain a `form.yml` file. These files are detected as form recipes and may be applied to groups and systems from the Web UI:

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

Editing Pillar Data in Uyuni

Uyuni requires a file called `form.yml`, to describe how formula data should look within the Web UI. `form.yml` is used by Uyuni to generate the desired form, with values editable by a user.

For example, the `form.yml` that is included with the locale-formula is placed in:

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

See part of the following locale-formula example:

```
# This file is part of locale-formula.
```

```
#
# Foobar is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# Foobar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with Foobar.  If not, see <http://www.gnu.org/licenses/>.

timezone:
  $type: group

  name:
    $type: select
    $values: ["CET",
              "CST6CDT",
              "EET",
              "EST",
              "EST5EDT",
              "GMT",
              "GMT+0",
              "GMT-0",
              "GMT0",
              "Greenwich",
              "HST",
              "MET",
              "MST",
              "MST7MDT",
              "NZ",
              "NZ-CHAT",
              "Navajo",
              "PST8PDT",
              "UCT",
              "UTC",
              "Universal",
              "W-SU",
              "WET",
              "Zulu",
              "Etc/GMT+1",
              "Etc/GMT+2",
              "Etc/GMT+3",
              "Etc/GMT+4",
              "Etc/GMT+5",
              "Etc/GMT+6",
              "Etc/GMT+7",
              "Etc/GMT+8",
              "Etc/GMT+9",
              "Etc/GMT+10",
              "Etc/GMT+11",
              "Etc/GMT+12",
              "Etc/GMT-1",
              "Etc/GMT-2",
              "Etc/GMT-3",
              "Etc/GMT-4",
              "Etc/GMT-5",
              "Etc/GMT-6",
              "Etc/GMT-7",
              "Etc/GMT-8",
              "Etc/GMT-9",
              "Etc/GMT-10",
              "Etc/GMT-11",
              "Etc/GMT-12",
              "Etc/GMT-13",
```

```
              "Etc/GMT-14",
              "Etc/GMT",
              "Etc/GMT+0",
              "Etc/GMT-0",
              "Etc/GMT0",
              "Etc/Greenwich",
              "Etc/UCT",
              "Etc/UTC",
              "Etc/Universal",
              "Etc/Zulu"
              ]
    $default: CET

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

`form.yml` contains additional information that describes how the form for a pillar should look for Uyuni. This information is contained in attributes that always start with a `$` sign.

Ignored Values

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.

The following are valid 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 following represent the supported types:

- `text`
- `password`
- `number`
- `url`
- `email`
- `date`
- `time`
- `datetime`
- `boolean`
- `color`
- `select`
- `group`

- `edit-group`
- `namespace`
- `hidden-group` (obsolete, renamed to `namespace`)

Text Attribute

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. The difference between `group` and `namespace` is `group` generates a visible border with a heading, and `namespace` shows nothing visually (and is only used to structure pillar data). The difference between `group` and `edit-group` is: `edit-group` allows to structure and restrict editable fields in a more flexible way. `edit-group` is a collection of items of the same kind; collections can have the following four "shapes":

- A list of primitive items
- A list of dictionaries
- A dictionary of primitive items
- A 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, and thus it simplifies complex and repeatable input structures. These, and also `itemName`, are optional. For an `edit-group` example, see Simple edit-group Example.

$default

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

$optional

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

$ifEmpty

The value to be used if the field is empty (because the user did not input any value). `ifEmpty` can only be used when `$optional` is `false` or not defined at all! If `$optional` is `true`, then `$ifEmpty` is ignored. In the following example, the `DP2` string would be used if user leaves the field empty:

```
displayName:
  $type: string
  $ifEmpty: DP2
```

$name

`$name` allows you to specify the name of a value that is shown in the form. If this value is not set, the pillar name is used and capitalized without underscores and dashes. You reference it in the same section with `${name}`.

$help and $placeholder

The `$help` and `$placeholder` attributes are used to give a user a better understanding of what the value should be.

- `$help` defines the message a user sees when hovering over a field
- `$placeholder` displays a gray placeholder text in the field

`$placeholder` may only be used with text fields like text, password, email or date. It does not make sense to add a placeholder if you also use `$default` as this will hide the placeholder.

$key

`$key` is applicable if the `edit-group` has the "shape" of a dictionary; you use it when the pillar data is supposed to be a dictionary. The `$key` attribute then determines the key of an entry in the dictionary. 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` allow you to specify the lowest and highest number the group can occur.

$itemName

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

$prototype

In an `edit-group`, `$prototype` is mandatory and allows to define default (or pre-filled) values for newly added members in the group.

$scope

`$scope` allows you to specify a hierarchy level at which a value may be edited. Possible values are `system`, `group`, and `readonly`.

The default `$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.

If using `$scope: group`, a value may only be edited for a group. On the system level you will be able to see the value, but not edit it.

The `$scope: readonly` option makes a field read-only. It can be used to show a user data which should be known, but should not be editable. This option only makes sense in combination with the `$default` attribute.

$visibleIf

`$visibleIf` allows you to show a field or group if a simple condition is met. A condition always looks similar to the following example:

```
some_group#another_group#my_checkbox == true
```

The left part of the above statement is the path to another value, and groups are separated by `$` signs. The middle section of the command 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 now 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, etc.

A check box should be structured like the following example:

```
some_group:
  $type: group

  another_group:
    $type: group

      my_checkbox:
        $type: boolean
```

Relative paths can be specified using prefix dots. One dot means sibling, 2 dots mean parent, etc. This is mostly useful for `edit-group`.

```
some_group:
  $type: group

  another_group:
    $type: group

    my_checkbox:
      $type: boolean

    my_text:
      $visibleIf: .my_checkbox

  yet_another_group:
    $type: group

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

By using multiple groups with the attribute, you can allow a user to select an option and show a completely different form, dependent upon the selected value.

Values from hidden fields may be merged into the pillar data and sent to the minion. 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

`$values` can only be used together with `$type`: select 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 alternatively:

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

Simple edit-group Example

See the following **edit-group** example:

```
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
```

After clicking **[Add]** for one time you will see `edit-group` Example in the Web UI filled with the default values. The formula itself is called `hd-partitions` and will appear as `Hd Partitions` in the Web UI.

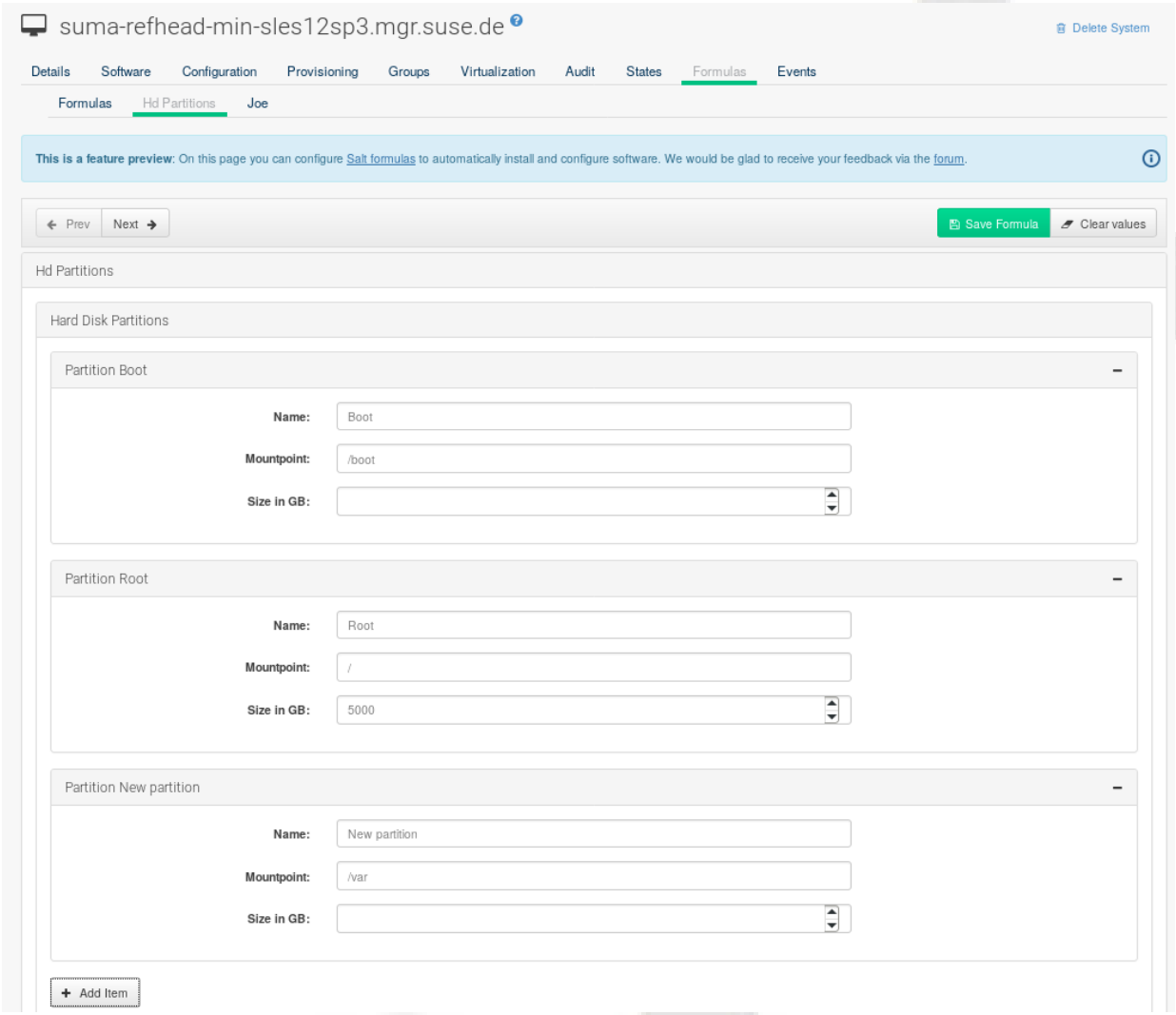

Figure 1. `edit-group` *Example in the Web UI*

To remove the definition of a partition click the minus symbol in the title line of an inner group. When form fields are properly filled confirm with clicking **[Save Formula]** in the upper right corner of the formula.

Writing Salt Formulas

Salt formulas are pre-written Salt states, which may be configured with pillar data. You can parametrize state files using Jinja. Jinja allows you to access pillar data by using the following syntax. This syntax works best when you are uncertain whether a pillar value exists as it will throw an error:

```
pillar.some.value
```

When you are sure a pillar exists you may also use the following syntax:

```
salt['pillar.get']('some:value', 'default value')
```

You may also replace the `pillar` value with `grains` (for example, `grains.some.value`) allowing access to grains.

Using data this way allows you to make a formula configurable. The following code snippet will install a package specified in the pillar `package_name`:

```
install_a_package:
  pkg.installed:
    - name: {{ pillar.package_name }}
```

You may 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 may want to have Salt manage your files (for example, configuration files for a program), and you can change these with pillar data. For example, the following snippet shows how you can manage a file using Salt:

```
/etc/my_program/my_program.conf:
  file.managed:
    - source: salt://my_state/files/my_program.conf
    - template: jinja
```

Salt will copy the file `salt-file_roots/my_state/files/my_program.conf` on the salt master to `/etc/my_program/my_program.conf` on the minion and template it with Jinja. This allows you to use Jinja in the file, exactly like shown above for states:

```
some_config_option = {{ pillar.config_option_a }}
```

Separating Data

It is often a good idea to separate data from a state to increase its flexibility and add re-usability value. This is often done by writing values into a separate file named `map.jinja`. This file should be placed within the same directory as your state files.

The following example will set `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 just using `data.value`.

You can also choose to override defined values from pillars (for example, by overriding `some.pillar.data.package` in the 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')) %}
```

After creating a map file like the above example, 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 also 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 %}
```

Formulas utilized with Uyuni should follow formula conventions listed in the official documentation:

- https://docs.saltstack.com/en/latest/topics/development/conventions/formulas.html

Uyuni Generated Pillar Data

When pillar data is generated (for example, after applying the highstate) the following external pillar script generates pillar data for packages, group ids, etc. and includes all pillar data for a system:

```
/usr/share/susemanager/modules/pillar/suma_minion.py
```

The process is executed as follows:

1. The `suma_minion.py` script starts and finds all formulas for a system (by checking the `group_formulas.json` and `server_formulas.json` files).
2. `suma_minion.py` loads the values for each formula (groups and from the system) and merges them with the highstate (default: if no values are found, a group overrides a system if $scope: group etc.).
3. `suma_minion.py` also includes a list of formulas applied to the system in a pillar named formulas. This structure makes it possible to include states. The top file (in this case specifically generated by the `mgr_master_tops.py` script) includes a state called formulas for each system. This includes the `formulas.sls` file located in:

   ```
   /usr/share/susemanager/formulas/states/
   ```

   The content looks similar to the following:

   ```
   include: {{ pillar["formulas"] }}
   ```

   This pillar includes all formulas, that are specified in pillar data generated from the external pillar script.

Formula Requirements

Formulas should be designed/created directly after a Uyuni installation, but if you encounter any issues check the following:

- The external pillar script (`suma_minion.py`) must include formula data.

- Data is saved to `/srv/susemanager/formula_data` and the `pillar` and `group_pillar` sub-directories. These should be automatically generated by the server.
- Formulas must be included for every minion listed in the top file. Currently this process is initiated by the `mgr_master_tops.py` script which includes the `formulas.sls` file located in:

```
/usr/share/susemanager/formulas/states/
```

  This directory must be a salt file root. File roots are configured on the salt-master (Uyuni) located in:

```
/etc/salt/master.d/susemanager.conf
```

Using Salt Formulas with Uyuni

The following procedure provides an overview on using Salt Formulas with Uyuni.

1. Official formulas may be installed as RPMs. Place the custom states within `/srv/salt/your-formula-name/` and the metadata (`form.yml` and `metadata.yml`) in `/srv/formula_metadata/your-formula-name/`. After installing your formulas they will appear in **Salt › Formula Catalog**.
2. To begin using a formula, apply it to a group or system. Apply a formula to a group or system by selecting the **System Details › Formulas** tab of a `System Details` page or `System Group`. From the **System Details › Formulas** page you can select any formulas you wish to apply to a group or system. Click the **[Save]** button to save your changes to the database.
3. After applying one or more formulas to a group or system, additional tabs will become available from the top menu, one for each formula selected. From these tabs you may configure your formulas.
4. When you have finished customizing your formula values you will need to apply the highstate for them to take effect. Applying the highstate will execute the state associated with the formula and configure targeted systems. You can use the **[Apply Highstate]** button from any formulas page of a group.
5. When a change to any of your values is required or you need to re-apply the formula state because of a failure or bug, change values located on your formula pages and re-apply the highstate. Salt will ensure that only modified values are adjusted and restart or reinstall services only when necessary.

This conclude your introduction to Salt Formulas. For additional information, see:

- https://docs.saltstack.com/en/latest/topics/development/conventions/formulas.html

Formulas

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

This section lists the primary formulas shipped with SUSE Manager for Retail and their configuration

options.

All the formulas in this section must be accurately configured for your SUSE Manager for Retail installation to function correctly. If you are unsure of the correct formula configuration details, run the `retail_branch_init` command before you begin to create the recommended formula configuration. You can then manually edit the formulas as required.

State and formula name collisions

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 the names of states and formulas to avoid name collisions.

Most formulas can be updated using the SUSE Manager Web UI. Once you have made changes to your formula, ensure you apply the highstate to propagate your changes to the appropriate services.

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.

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.

Procedure: Configuring Bind with Two Zones

1. Check the `Bind` formula, and click `Save`.
2. Navigate to the **Formulas** › **Bind** tab, and set these parameters for Zone 1:
    - In the `Config` section, select `Include Forwarders`.
    - In the `Name` field, enter the domain name of your branch network (for example: `branch1.example.org`).
    - In the `Type` field, select `master`.
3. Click `Add item` to save your changes.
4. Set these parameters for Zone 2:

- In the `Name` field, use the reverse zone for the configured IP range (for example: `1.168.192.in-addr.arpa`).
- In the `Type` field, select `master`

5. In the `Available Zones` section, use these parameters for Zone 1:
   - In the `Name` field, enter the domain name of your branch network (for example: `branch1.example.org`).
   - In the `File` field, type the name of your configuration file.
6. In the `Start of Authority (SOA)` section, use these parameters for Zone 1:
   - In the `Nameserver (Ns)` field, use the FQDN of the branch server (for example: `branchserver.branch1.example.org`).
   - In the `Contact` field, use the email address for the domain administrator.
   - Keep all other fields as their default values.
7. In the `Records` section, in subsection `A`, click **[Add Item]** and use these parameters to set up an A record for Zone 1:
   - In the `Hostname` field, use the hostname of the branch server (for example: `branchserver`).
   - In the `IP` field, use the IP address of the branch server (for example, `192.168.1.1`).
8. In the `Records` section, subsection `NS`, click **[Add Item]** and 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: `branchserver`).
9. In the `Records` section, subsection `CNAME`, click on **[Add Item]** and add the hostname of the branch server in each of these fields:
   - `tftp`
   - `ftp`
   - `dns`
   - `dhcp`
   - `salt`. The `salt` CNAME should be the FQDN of the branch server's external interface for proxy functionality to work correctly.
10. 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, `192.168.1.1/24`).

- For Zones should specify the domain name of your branch network (for example, branch1.example.org).

11. Click **[Save Formula]** to save your configuration.
12. Apply the highstate.

Reverse name resolution on terminals might not work for networks that are inside one of these IPv4 private address ranges:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

If you encounter this problem, go to the Options section of the Bind formula, and click **[Add item]**: * In the Options field, enter empty-zones-enable. * In the Value field, select No.

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.

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 SUSE Manager 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, which are configured through SUSE Manager for Retail formulas in the SUSE Manager Web UI.

Procedure: Setting up a branch server with a dedicated LAN

1. In the SUSE Manager Web UI, open the details page for the branch server, and navigate to the Formulas tab.
2. In the Branch Network section, set these parameters:
   - Keep Dedicated NIC checked
   - In the NIC field, enter the name of the network device that is connected to the internal LAN.
   - In the IP field, enter the static IP address to be assigned to the branch server on the internal LAN.

- In the `Netmask` field, enter the network mask of the internal LAN.

3. Check `Enable Route` if you want the branch server to route traffic from internal LAN to WAN.
    - Check `Enable NAT` if you want the branch server to convert addresses from internal LAN to WAN.
    - Select the `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.
4. Click **[Save]** to save your changes.
5. Apply the highstate.

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 SUSE Manager server as well as the terminals.

This configuration allows for the branch server to provide DNS, TFTP, PXE and FTP services to terminals, which are configured through SUSE Manager for Retail formulas in the SUSE Manager Web UI. Optionally, the branch server can also provide DHCP services in this configuration.

If DHCP services are not provided by the branch server, ensure that your external DHCP configuration is set correctly: * The `next-server` option must point to the branch server for PXE boot to work * The `filename` option must correctly identify the network boot program (by default, this is `/boot/pxelinux`) * The `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

1. In the SUSE Manager Web UI, open the details page for the branch server, and navigate to the `Formulas` tab.
2. In the `Branch Network` section, set these parameters:
    - Keep `Dedicated NIC` unchecked
    - Select which services to enable on the branch server's firewall. Ensure you include DNS, TFTP and FTP services.
    - Select the `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.
3. Click **[Save]** to save your changes.
4. Apply the highstate.

DHCPd Formula

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

Procedure: Configuring DHCP

1. In the SUSE Manager Web UI, open the details page for the branch server, and navigate to the Formulas tab.
2. Select the `Dhcpd` formula, and click **[Save]**.
3. Navigate to the **Formulas › Dhcpd** tab, and set these parameters:
    - In the `Domain Name` field, enter the domain name for the branch server (for example: `branch1.example.com`).
    - In the `Domain Name Server` field, enter either the IP address or resolvable FQDN of the branch DNS server (for example: `192.168.1.1`).
    - In the `Listen Interfaces` field, enter the name of the network interface used to connect to the local branch network (for example: `eth1`).
4. Navigate to the `Network Configuration (subnet)` section, and use these parameters for Network1:
    - In the `Network IP` field, enter the IP address of the branch server network (for example: `192.168.1.0`).
    - In the `Netmask` field, enter the network mask of the branch server network (for example: `255.255.255.0`).
    - In the `Domain Name` field, enter the domain name for the branch server network (for example: `branch1.example.com`).
5. In the `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: `192.168.1.51`).
    - In the second input box, set the upper bound of the IP range (for example: `192.168.1.151`).
6. In the `Broadcast Address` field, enter the broadcast IP address for the branch network (for example: `192.168.1.255`).
7. In the `Routers` field, enter the IP address to be used by routers in the branch server network (for example: `192.168.1.1`).
8. In the `Next Server` field, enter the hostname or IP address of the branch server (for example: `192.168.1.1`).
9. In the `Filename` field, keep the default value of `/boot/pxelinux.0`.
10. Click **[Save Formula]** to save your configuration
11. Apply the highstate.

PXE Formula

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

Procedure: Configuring PXE booting

1. In the SUSE Manager Web UI, open the details page for the branch server, and navigate to the `Formulas` tab.
2. Select the `Pxe` formula, and click `Save`.
3. Navigate to the **Formulas › Pxe** tab, and set these parameters:
    - In the `Kernel filename` field, keep the default value.
    - In the `Initrd filename` field, keep the default value.
    - In the `Kernel commandline parameters` field, keep the default value.
    - In the `PXE root directory` field, enter the path to the saltboot directory (for example, `/srv/saltboot`).
    - In the `Branch id` field, type a name to use as a branch identifier (for example: `Branch0001`). Use only alphanumeric characters for the branch identifier.
4. Click `Save Formula` to save your configuration
5. Apply the highstate.

Saltboot Formula

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

Saltboot formula is meant to be used as a group formula. Enable and configure saltboot formula for hardware type groups.

Procedure: Configuring the hardware type group with saltboot

1. Open the details page for your new hardware type group, and navigate to the `Formulas` tab.
2. Select the `saltboot-formula` and click **[Save]**.
3. Navigate to the new **Formulas › Saltboot** tab.
4. In the `Disk 1` section, set these parameters:
    - In the `Disk symbolic ID` field, enter a custom name for the disk (for example, `disk1`).
    - In the `Device type` field, select `DISK`.
    - In the `Disk device` field, select the device that corresponds to the device name on the target machine (for example, `/dev/sda`).
    - In the `RAID level` field, leave it empty.
    - In the `Disk Label` field, select `gpt`.

5. In the `Partition` section, set these parameters for `Partition 1`:
    - In the `Partition symbolic ID` field, enter a custom name for the partition (for example, `p1`).
    - In the `Partition size` field, specify a size for the partition in Mebibytes (MiB).
    - In the `Device mount point` field, select a location to mount the partition (for example, `/data`).
    - In the `Filesystem format` field, select your preferred format (for example, `xfs`).
    - In the `OS Image to deploy` field, leave it empty.
    - In the `Partition encryption password` field, enter a password if you want to encrypt the partition.
    - In the `Partition flags` field, leave it empty.
6. In the `Partition` section, set these parameters for `Partition 2`:
    - In the `Partition symbolic ID` field, enter a custom name for the partition (for example, `p2`).
    - In the `Partition size` field, specify a size for the partition in Mebibytes (MiB).
    - In the `Device mount point` field, leave it empty.
    - In the `Filesystem format` field, select `swap`.
    - In the `OS Image to deploy` field, leave it empty.
    - In the `Partition encryption password` field, enter a password if you want to encrypt the partition.
    - In the `Partition flags` field, select `swap`.
7. In the `Partition` section, set these parameters for `Partition 3`:
    - In the `Partition symbolic ID` field, enter a custom name for the partition (for example, `p3`).
    - In the `Partition size` field, leave it empty. This will ensure the partition uses up all remaining space.
    - In the `Device mount point` field, select `/`.
    - In the `Filesystem format` field, leave it empty.
    - In the `OS Image to deploy` field, enter the name of the image to deploy.
    - In the `Image version` field, leave it empty. This will ensure you use the latest available version.
    - In the `Partition encryption password` field, enter a password if you want to encrypt the partition.

- In the `Partition flags` field, leave it empty.

8. Click **[Save Formula]** to save your formula.

TFTPd Formula

The TFTPd formula is used to configure the TFTP service on the branch server.

Procedure: Configuring TFTP

1. In the SUSE Manager Web UI, open the details page for the branch server, and navigate to the `Formulas` tab.
2. Select the `Tftpd` formula, and click **[Save]**.
3. Navigate to the **Formulas › Tftpd** tab, and set these parameters:
   - In the `Internal Network Address` field, enter the IP address of the branch server (for example: `192.168.1.1`).
   - In the `TFTP Base Directory` field, enter the path to the saltboot directory (for example, `/srv/saltboot`).
   - In the `Run TFTP Under User` field, enter `saltboot`.
4. Click **[Save Formula]** to save your configuration.
5. Apply the highstate.

VsFTPd Formula

The VsFTPd formula is used to configure the FTP service on the branch server.

Procedure: Configuring VsFTPd

1. In the SUSE Manager Web UI, open the details page for the branch server, and navigate to the `Formulas` tab.
2. Select the `Vsftpd` formula, and click **[Save]**.
3. Navigate to the **Formulas › Vsftpd** tab, and set these parameters:
   - In the `Internal Network Address`, enter IP address of branch server (for example: `192.168.1.1`).
   - All other fields can retain their default values.
4. Click **[Save Formula]** to save your configuration
5. Apply the highstate.

Salt Formulas Coming with SUSE Manager

For general information, see the `Salt Formulas installation and usage instructions` at https://docs.saltstack.com/en/latest/topics/development/conventions/formulas.html.

Locale

The locale formula allows setting `Timezone` and `[guimenu]`Keyboard and Language`.

Domain Name System (Bind)

With the bind formula you set up and configure a Domain Name System (DNS) server. For technical information about the bind formula and low-level pillar data, see the `README.rst` file on the Uyuni server: `/usr/share/susemanager/formulas/metadata/bind/README.rst`.

DNS is needed to resolve the domain names and host names into IP addresses. For more information about DNS, see the SLES Administration Guide, Services, The Domain Name System.

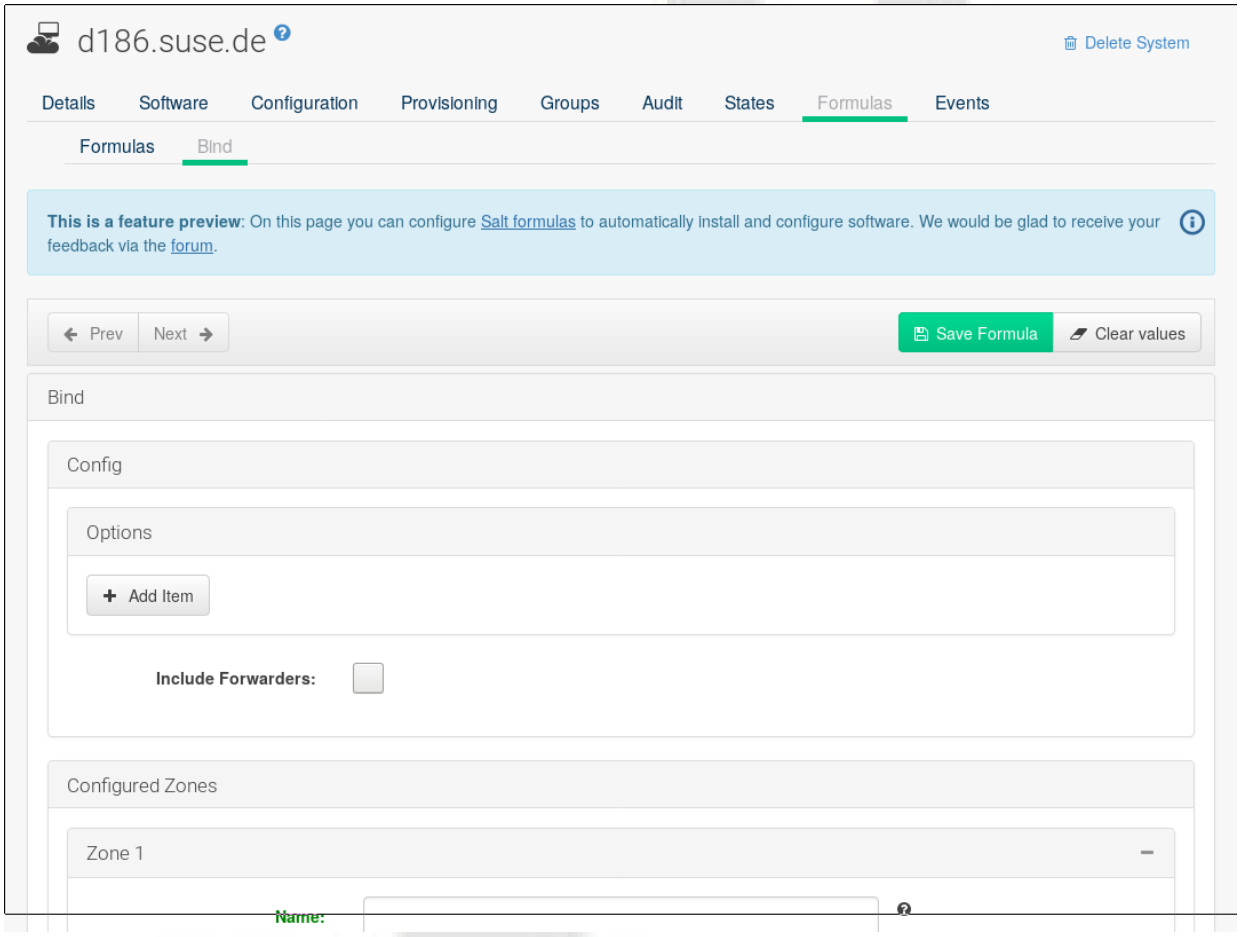

Figure 2. Bind Formula

In the `Config` group you can set arbitrary options such as `directory` where are the zone data files (usually `/var/lib/named/`) or `forwarders`. Click **[Add Item]** to provide more Key/Value fields for configuration.

Check `Include Forwarders` if you want to rely on an external DNS server if your DNS is down (or is otherwise not able to resolve an address).

At least, you will configure one zone. In `Configured Zones` define your zone; for example,

`example.com`. Then in `Available Zones` configure this zone: as `Name` enter your zone (in this case `example.com`) and the `File` to which this configuration should be written (`example.com.txt`). Enter the mandatory `SOA` record (start of authority), and the A, NS, and CNAME `Records` you need.

On the other hand, if no `records` entry exists, the zone file is not generated by this state rather than taken from `salt://zones`. For how to overwrite this URL, see `pillar.example`.

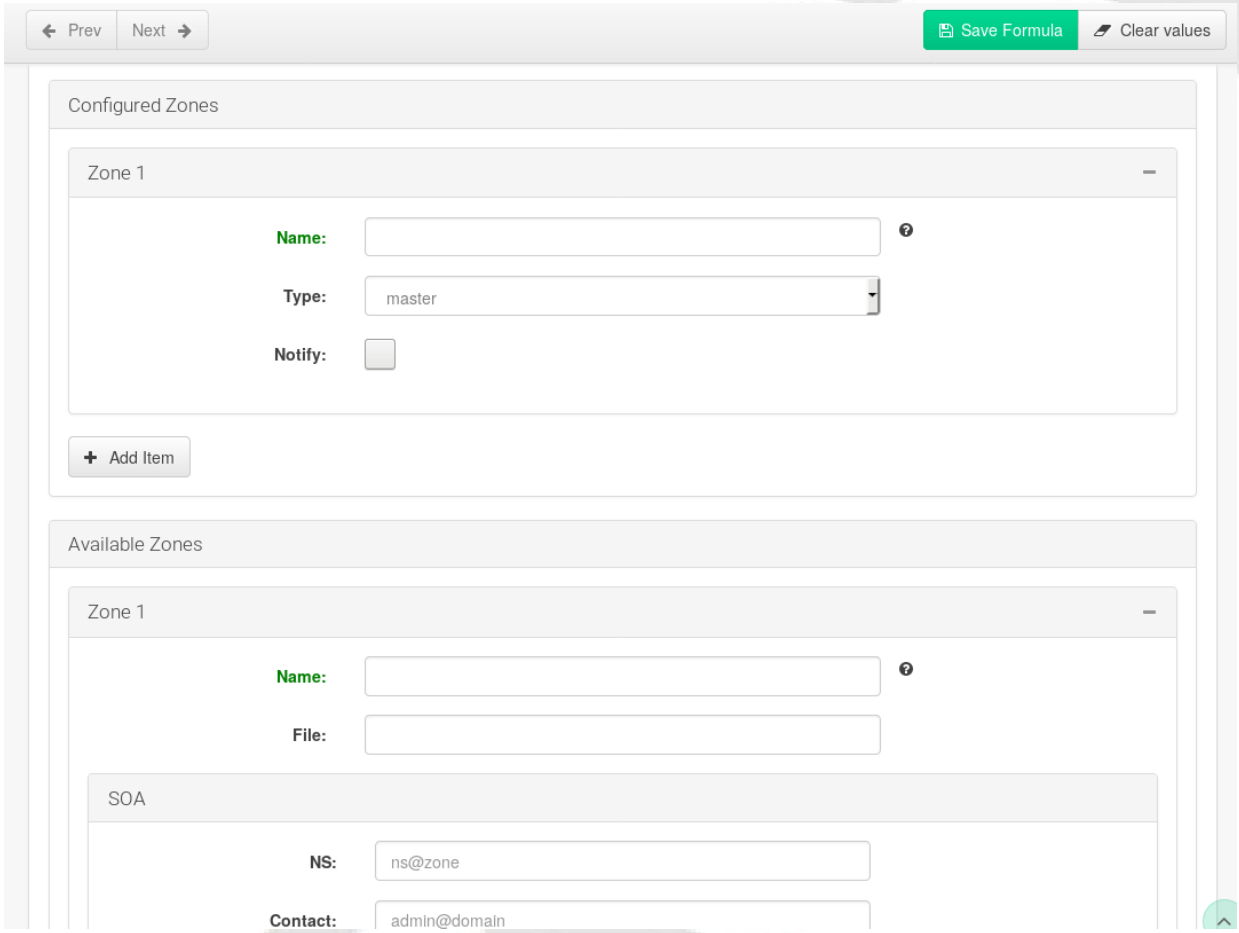

Figure 3. bind-02-zones

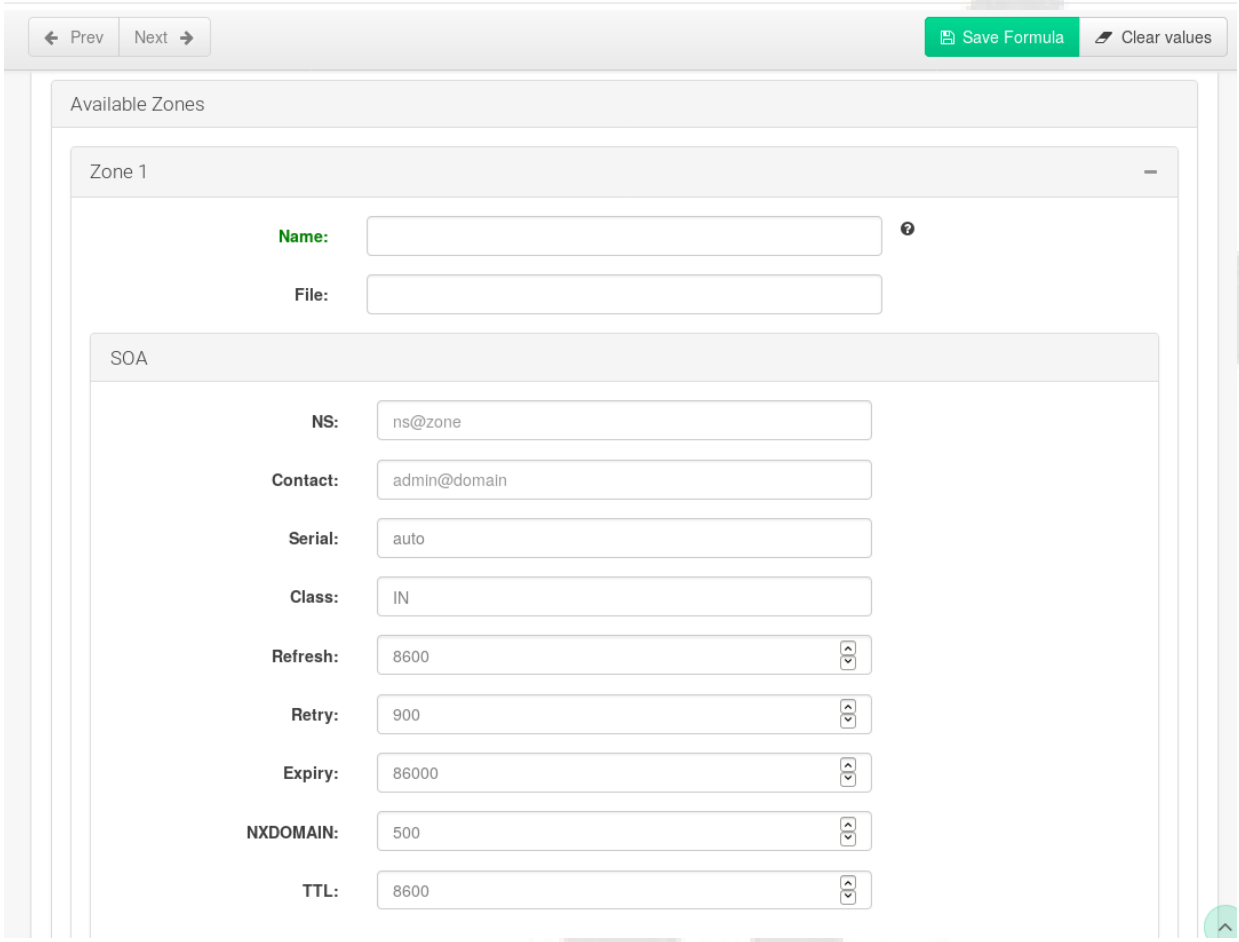

Figure 4. bind-03-records

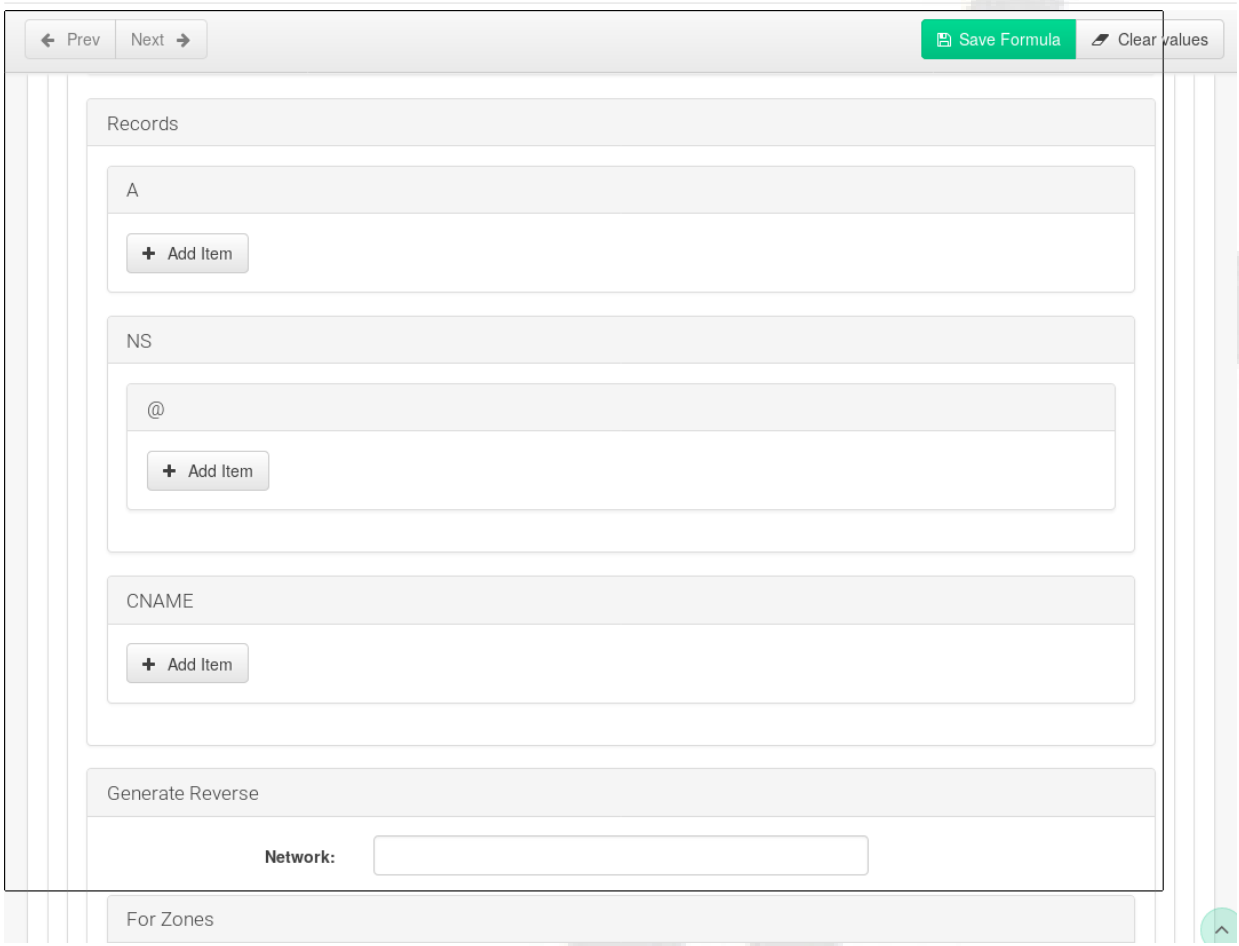

Figure 5. bind-03-records2

In `Generate Reverse`, and define reverse mapping and for which zones:

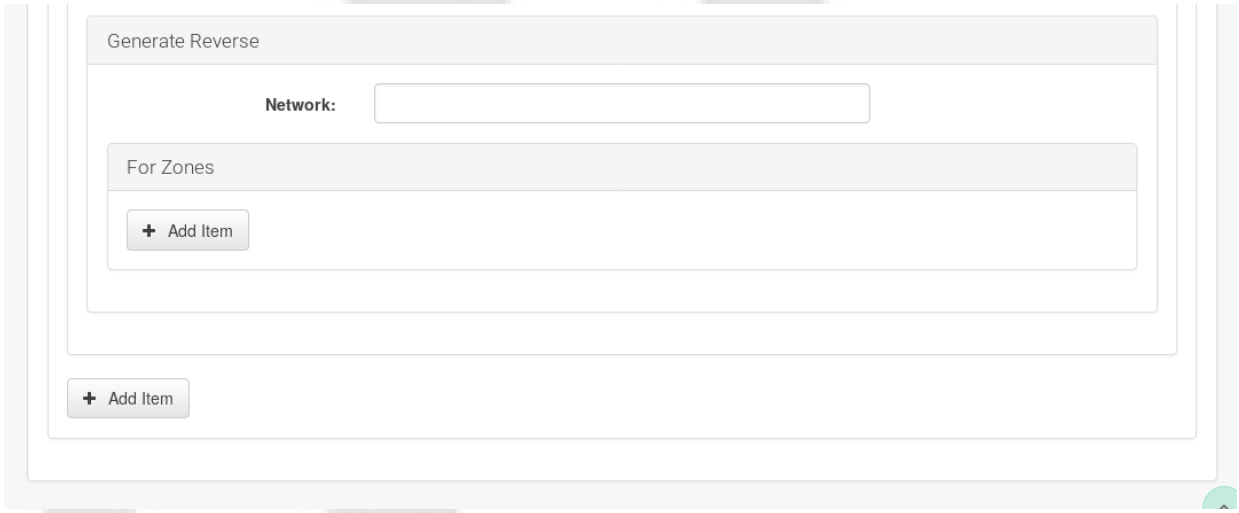

Figure 6. bind-04-reverse

When saved, data is written to `/srv/susemanager/formula_data/pillar/<salt-minion.example.com>_bind.json`.

If you apply the highstate (**System Details › States › Highstate**), it first ensures that `bind` and all required packages will get installed. Then it will start the DNS service (`named`).

Dhcpd

With the dhcpd formula you set up and configure a DHCP server (Dynamic Host Configuration Protocol). For technical information about the dhcpd formula and low-level pillar data, see the Pillar example file `/usr/share/susemanager/formulas/metadata/dhcpd/pillar.example`.

DHCP is needed to define network settings centrally (on a server) and let clients retrieve and use this information for local host configuration. For more information about DHCP, see the SLES Administration Guide, Services, DHCP.

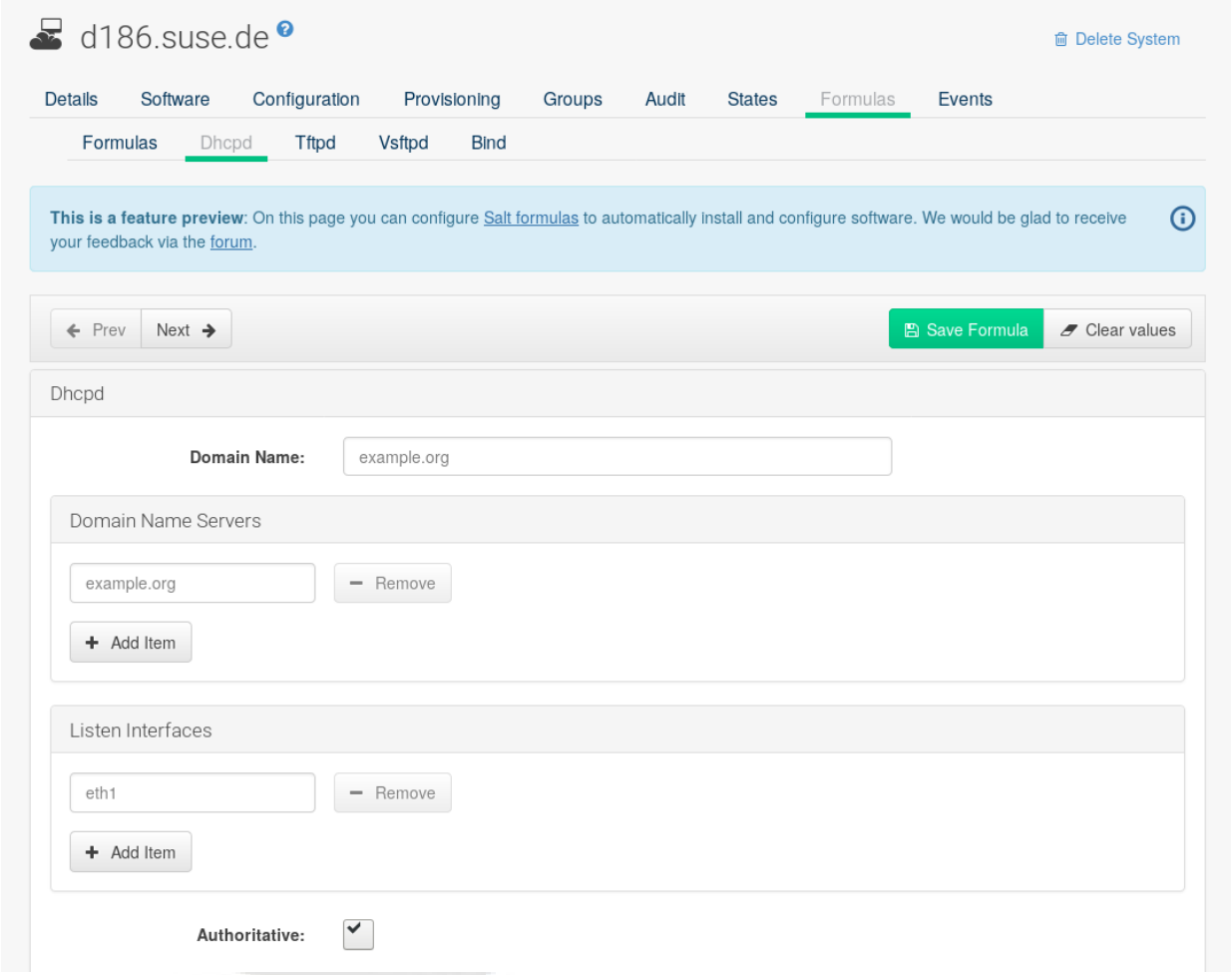

Figure 7. dhcpd formula

Domain Name.

`Domain Name Servers`. One or more Domain Name Service (DNS) servers.

On which interface(s) the DHCP server should listen (`Listen interfaces`). Set option for this interface: Authoritative: Max Lease Time: Default Lease Time:

Next is at least one network in the `Network configuration (subnet)` group (with IP address, netmask, etc.). You define every network with `Dynamic IP range`, `Routers`, and `Hosts with static IP addresses (with defaults from subnet)` (optionally).

And finally `Hosts with static IP addresses (with global defaults)`.

If you apply the highstate (**System Details › States › Highstate**), it first ensures that `dhcp-server` and all required packages will get installed. Then it will start the DHCP service (`dhcpd`).

Tftpd

With the tftpd formula you set up and configure a TFTP server (Trivial File Transfer Protocol). A TFTP server is a component that provides infrastructure for booting with PXE.

For more information about setting up TFTP, see the SLES Deployment Guide, Preparing Network Boot Environment, Setting Up a TFTP Server.

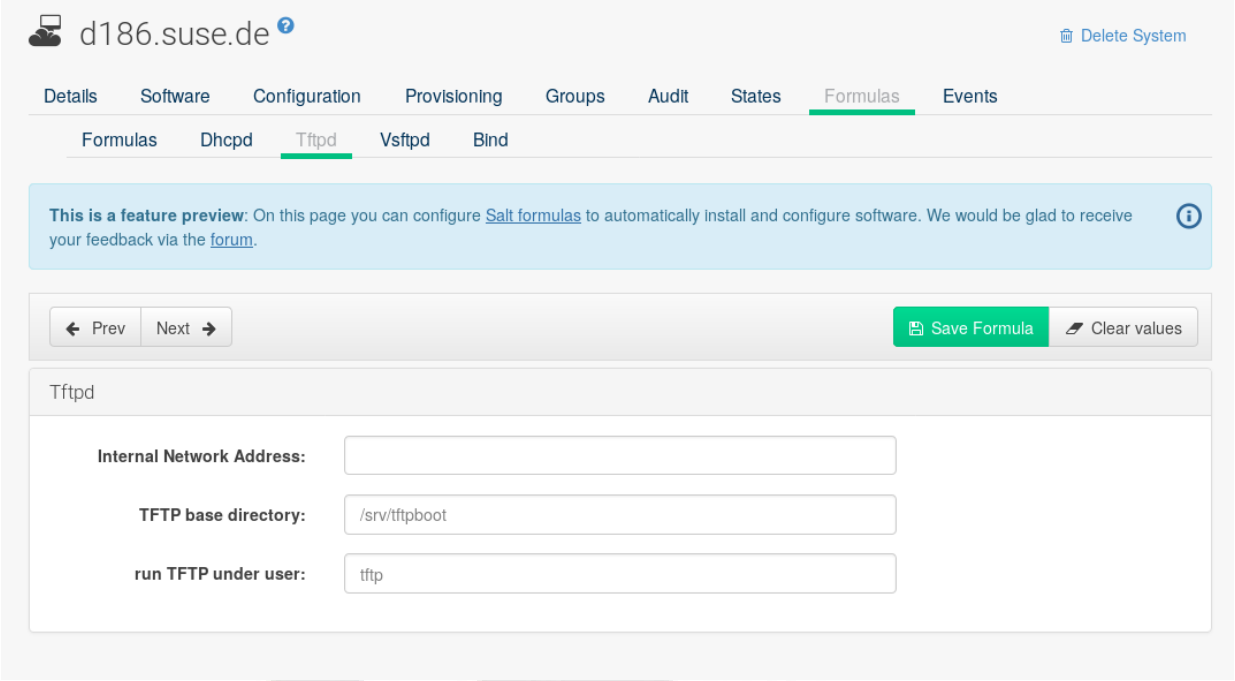

Figure 8. tftpd formula

For setting up a TFTP server, specify the `Internal Network Address`, `TFTP base directory` (default: `/srv/tftpboot`), and `run TFTP under user` (default: `sftp`).

If you apply the highstate (**System Details › States › Highstate**), it first ensures that `atftp` and all required packages will get installed. Then it will start TFTP (`atftpd`).

Vsftpd

With the vsftpd formula you set up and configure Vsftpd. Vsftpd is an FTP server or daemon, written with security in mind. "vs" in its name stands for "Very Secure".

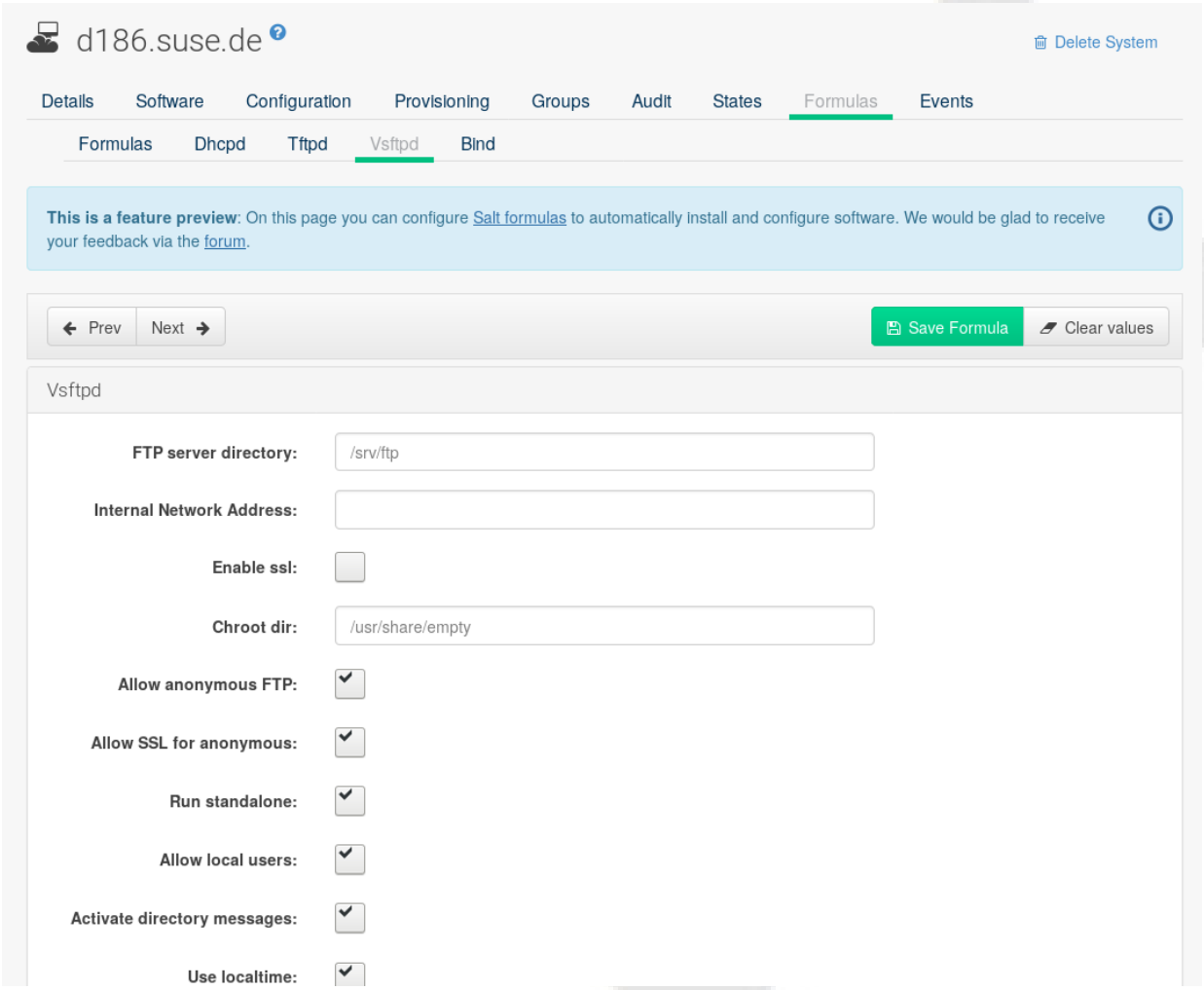

Figure 9. vsftpd formula

For configuring a VSFTP server, specify the settings and options in the Vsftpd formula. There are settings such as `FTP server directory`, `Internal Network Address` `Enable ssl`, etc.

If you apply the highstate (**System Details › States › Highstate**), it first ensures that `vsftpd` and all required packages will get installed. Then it will start the VSFTP service (`vsftpd`).

For more information about setting up and tuning Vsftpd, see the documentation coming with the `vsftpd` package (`/usr/share/doc/packages/vsftpd/` when the package is installed).

Install the SUSE Manager Locale Formula

The following section provides guidance on installing and using SUSE provided Salt formulas.

Procedure: Installing the Locale Formula

1. Install the locale formula with:

```
zypper install locale-formula
```

This installs the package contents to `/usr/share/susemanager/formulas/{metadata,states}`

2. After installing the RPM, log in to the Uyuni Web UI.
3. Browse to the **Main Menu › System Details** page of any minion you would like to apply the formula to.
4. On the **Main Menu › System Details** page you will see a new **[Formulas]** tab. Select it to view a list of installed formulas.
5. From the **[Formulas]** list select **Formulas › Locale** and click **[Save]**.
6. A new tab will appear next to the **Formula › Locale** subtab. Select the new **Formulas › Locale** tab.
7. The **Formalas › Locale** tab contains options for setting the language, keyboard layout, timezone, and whether hardware clock is set to UTC. Select the desired options and click **[Save]**.
8. Run the following command to verify pillar settings. The output has been truncated.

```
salt '$your_minion' pillar.items
```

```
...
   keyboard_and_language:
       ----------
       keyboard_layout:
           English (US)
       language:
           English (US)
   machine_password:
       foobar
   mgr_server:
       manager_server
   org_id:alt '$your_minion_here'
       1
   timezone:
       ----------
       hardware_clock_set_to_utc:
           True
       name:
           CET
    ...
```

9. Apply this state to your minion by applying the highstate from the command line with:

```
salt '$your_minion' state.highstate
```

You can also apply the highstate from the previous formula tab from the Uyuni Web UI by selecting **System Details › States** and clicking **[Apply Highstate]**.