SUSE Manager Architecture

The SUSE Manager Team

March 02, 2019

Table of Contents

Introduction

ROOT:partial$entities.adoc

Component Legend

ROOT:partial$entities.adoc

These diagram components will be used in the following sections explaining the architecture of SUSE Manager. Components in SUSE Manager can communicate in three ways:

- One way
- Two way
- Scheduled (time based)

Types of Components

One Way

Components that communicate in only one direction are represented by:

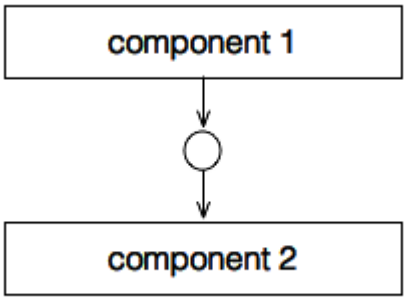


Figure 1. One way communication between components

Two Way

Components that communicate in both directions are represented by:

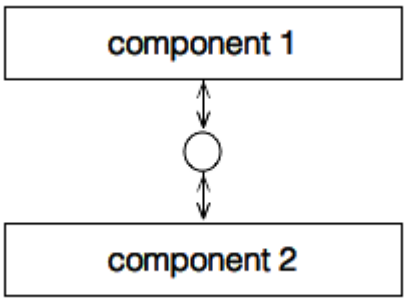


Figure 2. Two way communication between components

Database Connections

A component that reads and writes to the database communicates in both directions are represented by:

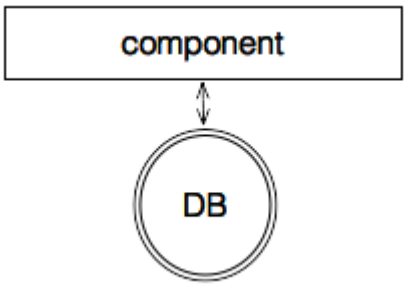


Figure 3. Two way communication between a component and the database(read and write)

Scheduled (Time based)

Components that run on a schedule are represented by:

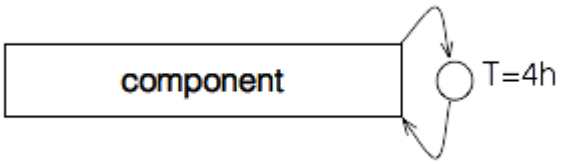


Figure 4. Component that runs on a schedule

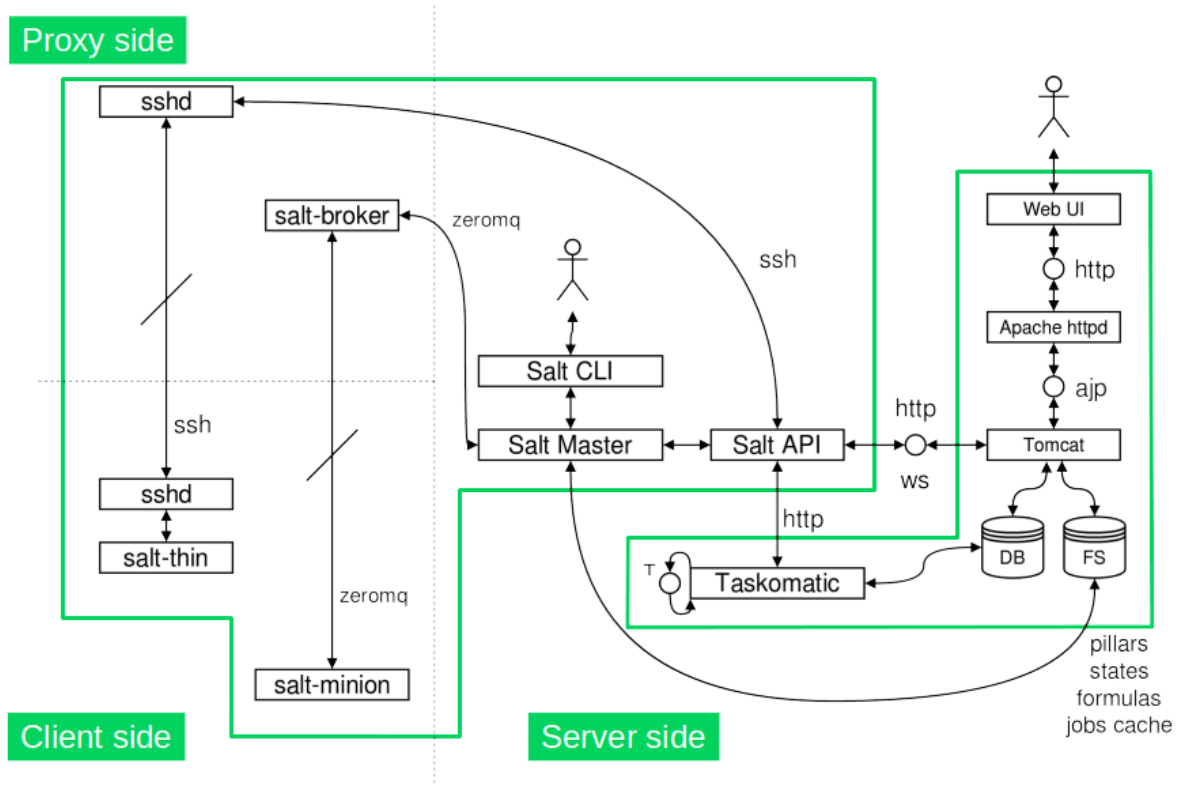
Salt

Salt Architecture

ROOT:partial$entities.adoc

Some description...

Salt Stack Diagram



Core Salt Components

ROOT:partial$entities.adoc

Comming soon...

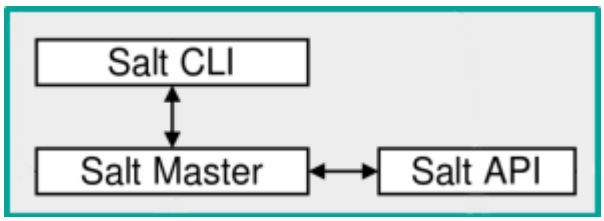


Figure 5. Salt Core

Salt Data and SUSE Manager

ROOT:partial$entities.adoc

Salt Global Static Data

Global static data **should not** be customized or edited by SUSE Manager users. This data is generated by the server.

Table 1. Salt Global Static Data

Directory	Function
/usr/share/susemanager/salt/	custom modules, states, grains
/usr/share/susemanager/pillar_data/	global pillar data
/usr/share/susemanager/formulas/	formulas

Generated Data Per Minion

Generated data for minions **should not** be customized or edited by SUSE Manager users. This data is generated by the server.

Table 2. Salt Generated Data Per Minion

Directory	Function
/srv/susemanager/pillar_data/	custom modules, states, grains
/usr/share/susemanager/pillar_data/	global pillar data
/srv/susemanager/formulas_data/	formulas

Custom Salt Data

The following directories are reserved for use by users **and should be** customized and edited by SUSE Manager users. The custom salt data place here will be calculated and combined with the content generated listed above when running a highstate.

Table 3. Salt Generated Data Per Minion

Directory	Function
/srv/salt/	user defined custom modules, states, grains
/srv/pillar/	user defined global pillar data
/srv/formula_metadata	user defined formulas

Salt Contact Methods

ROOT:partial$entities.adoc

Choosing a Contact Method for Salt

SUSE Manager provides several methods for communication between client and server. All commands that the SUSE Manager server sends to its clients will be routed through one of these contact methods.

The contact method you select for Salt will depend on your network infrastructure. The following sections provide a starting point for selecting a method which best suits your network environment.

- Salt Pull
- Salt SSH Push
- Salt SSH Push and Tunnel

Salt Pull

ROOT:partial$entities.adoc

Salt SSH Push

ROOT:partial$entities.adoc

Salt SSH Push is intended to be used in environments where your Salt clients cannot reach the SUSE Manager server directly to regularly checking in and, for example, fetch package updates.

Push via SSH

This feature is not related to Push via SSH for the traditional clients. For Push via SSH, see xref:bp.contact.methods.ssh.push[Salt SSH Push].

Overview

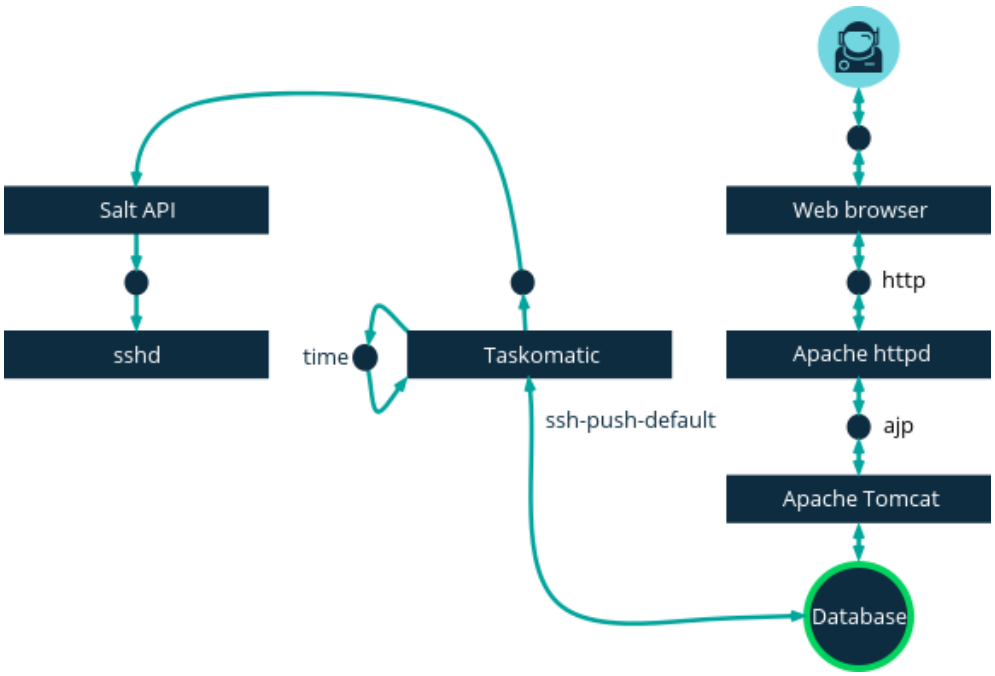


Figure 6. Push via Salt SSH Contact Method

Salt provides "Salt SSH" (`salt-ssh`), a feature to manage clients from a server. It works without installing Salt related software on clients. Using Salt SSH there is no need to have minions connected to the Salt master. Using this as a SUSE Manager connect method, this feature provides similar functionality for Salt clients as the traditional Push via SSH feature for traditional clients.

This feature allows:

- Managing Salt entitled systems with the Push via SSH contact method using Salt SSH.
- Bootstrapping such systems.

Requirements

- SSH daemon must be running on the remote system and reachable by the `salt-api` daemon (typically running on the SUSE Manager server).
- Python must be available on the remote system (Python must be supported by the installed Salt). Currently: python 2.6.

Unsupported Systems

Red Hat Enterprise Linux and CentOS versions ⇐ 5 are not supported because they do not have Python 2.6 by default.

Bootstrapping

To bootstrap a Salt SSH system, proceed as follows:

1. Open the **Bootstrap Minions ›] dialog in the Web UI (menu:Systems[Bootstrapping**).
2. Fill out the required fields. Select an **Activation Key ›] with the menu:Push via SSH[** contact method configured. For more information about activation keys, see: xref:ref.webui.systems.activ-keys.
3. Check the **Manage system completely via SSH** option.
4. Confirm with clicking the **Bootstrap** button.

Now the system will be bootstrapped and registered in SUSE Manager. If done successfully, it will appear in the **Systems** list.

Configuration

There are two kinds of parameters for Push via Salt SSH:

- Bootstrap-time parameters - configured in the **Bootstrapping** page:
  - Host
  - Activation key
  - Password - used only for bootstrapping, not saved anywhere; all future SSH sessions are authorized via a key/certificate pair
- Persistent parameters - configured SUSE Manager-wide:
  - sudo user - same as in bp.contact.methods.ssh.push.sudo.

Action Execution

The Push via Salt SSH feature uses a taskomatic job to execute scheduled actions using `salt-ssh`. The taskomatic job periodically checks for scheduled actions and executes them. While on traditional clients with SSH push configured only `rhn_check` is executed via SSH, the Salt SSH push job executes a complete `salt-ssh` call based on the scheduled action.

Known Limitation

- OpenSCAP auditing is not available on Salt SSH minions.
- Beacons do not work with Salt SSH.

- Installing a package on a system using `zypper` will not invoke the package refresh.
- Virtual Host functions (for example, a host to guests) will not work if the virtual host system is Salt SSH-based.

For More Information

For more information, see

- https://wiki.microfocus.com/index.php/SUSE_Manager/SaltSSHServerPush
- https://docs.saltstack.com/en/latest/topics/ssh/

Salt SSH Push & Tunnel

ROOT:partial$entities.adoc

Boostrapping UI

ROOT:partial$entities.adoc

Onboarding and Registration

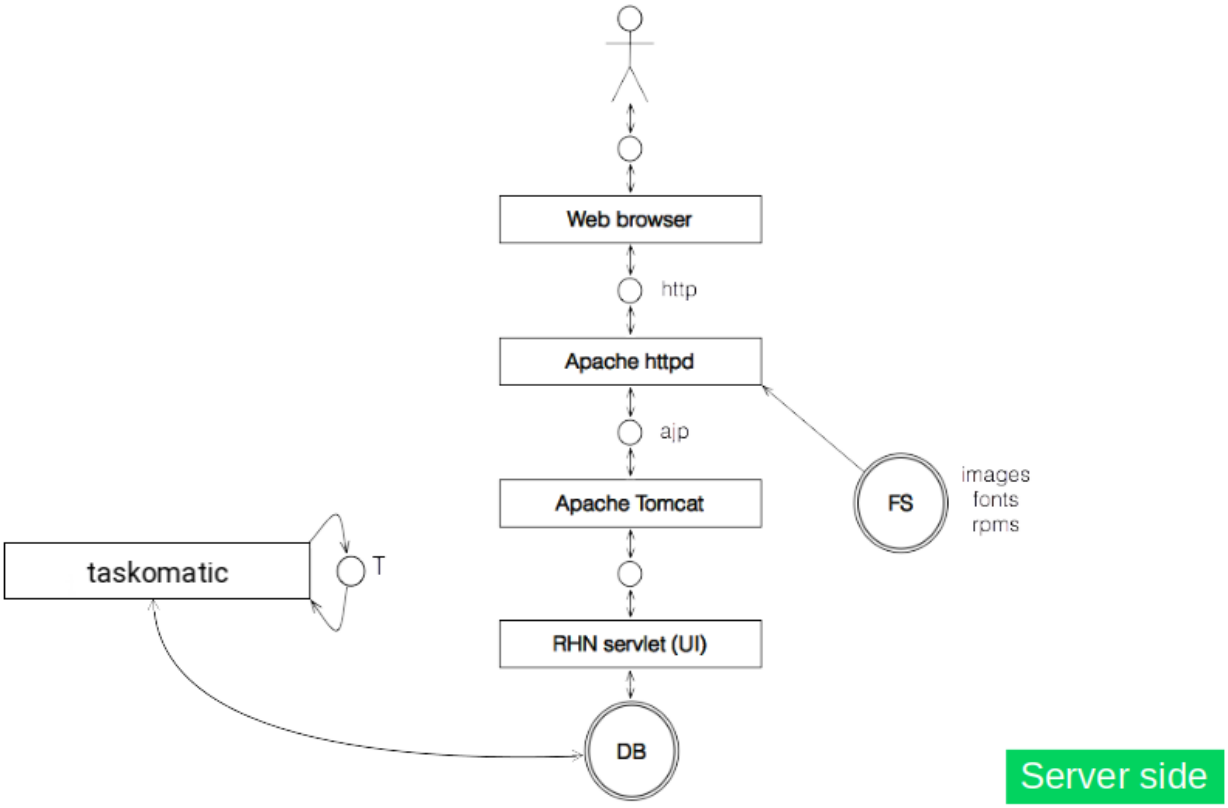
ROOT:partial$entities.adoc

The Traditional Stack

Traditional Architecture

ROOT:partial$entities.adoc

The Traditional Stack



Traditional Contact Methods

ROOT:partial$entities.adoc

Selecting a Contact Method

SUSE Manager provides several methods for communication between client and server. All commands that the SUSE Manager server sends to its clients will be routed through one of these contact methods.

The contact method you select will depend on your network infrastructure. The following sections provide a starting point for selecting a method which best suits your network environment.

Traditional Contact Method (rhnsd)

ROOT:partial$entities.adoc

The Default Contact Method

The SUSE Manager **rhnsd** daemon runs on client systems and periodically connects with SUSE Manager to check for new updates and notifications. The daemon, which runs in the background, is started by **rhnsd.service**. By default, it will check every 4 hours for new actions, therefore it may take some time for your clients to begin updating after actions have been scheduled for them.

To check for updates, **rhnsd** runs the external **mgr_check** program located in `/usr/sbin/`. This is a small application that establishes the network connection to SUSE Manager. The SUSE Manager daemon does not listen on any network ports or talk to the network directly. All network activity is done via the **mgr_check** utility.

Auto accepting (EULAs)

When new packages or updates are installed on the client using SUSE Manager, any end user licence agreements (EULAs) are automatically accepted. To review a package EULA, open the package detail page in the Web UI.

This figure provides an overview of the default **rhnsd** process path. All items left of the **Python XMLRPC server** block represent processes running on an SUSE Manager client.

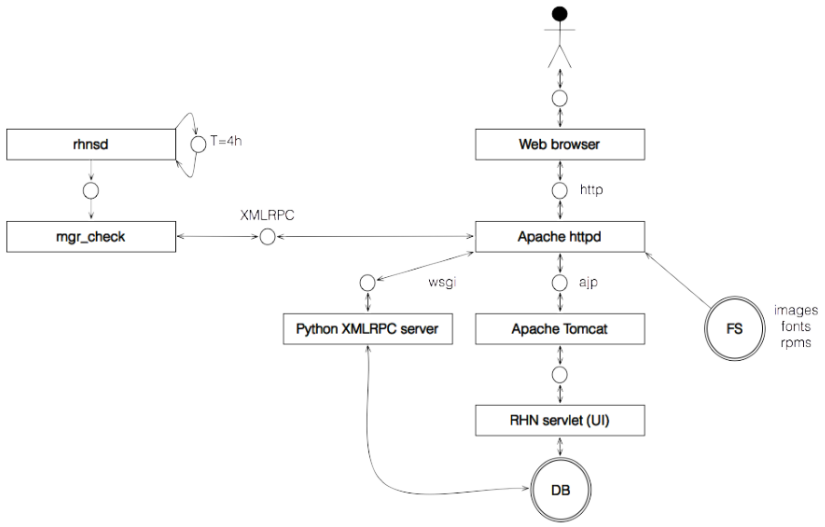


Figure 7. rhnsd Contact Method

Configuring SUSE Manager rhnsd Daemon

The SUSE Manager daemon can be configured by editing the file on the client:

```
/etc/sysconfig/rhn/rhnsd
```

This is the configuration file the rhnsd initialization script uses. An important parameter for the daemon is its check-in frequency. The default interval time is four hours (240 minutes). If you modify the configuration file, you must as root restart the daemon with:

```
systemctl rhnsd restart
```

Minimum Allowed Check-in Parameter

The minimum allowed time interval is one hour (60 minutes). If you set the interval below one hour, it will change back to the default of 4 hours (240 minutes).

Viewing rhnsd Daemon Status

As the root you may view the status of rhnsd by typing the command:

```
systemctl status rhnsd
```

Traditional Contact Method (osad)

ROOT:partial$entities.adoc

OSAD is an alternative contact method between SUSE Manager and its clients. By default, SUSE Manager uses `rhnsd`, which contacts the server every four hours to execute scheduled actions. OSAD allows registered client systems to execute scheduled actions immediately.

OSAD has several distinct components:

- The `osa-dispatcher` service runs on the server, and uses database checks to determine if clients need to be pinged, or if actions need to be executed.
- The `osad` service runs on the client. It responds to pings from `osa-dispatcher` and runs `mgr_check` to execute actions when directed to do so.
- The `jabberd` service is a daemon that uses the `XMPP` protocol for communication between the client and the server. The `jabberd` service also handles

authentication.

- The `mgr_check` tool runs on the client to execute actions. It is triggered by communication from the `osa-dispatcher` service.

The `osa-dispatcher` periodically runs a query to check when clients last showed network activity. If it finds a client that has not shown activity recently, it will use `jabberd` to ping all `osad` instances running on all clients registered with your SUSE Manager server. The `osad` instances respond to the ping using `jabberd`, which is running in the background on the server. When the `osa-dispatcher` receives the response, it marks the client as online. If the `osa-dispatcher` fails to receive a response within a certain period of time, it marks the client as offline.

When you schedule actions on an OSAD-enabled system, the task will be carried out immediately. The `osa-dispatcher` periodically checks clients for actions that need to be executed. If an outstanding action is found, it uses `jabberd` to execute `mgr_check` on the client, which will then execute the action.

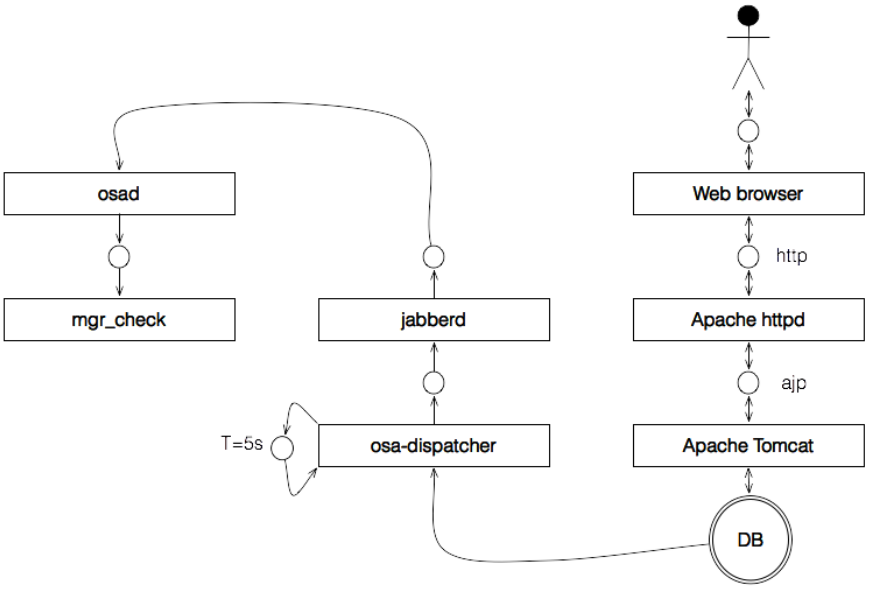


Figure 8. osad Contact Method

Enabling and Configuring OSAD

This section covers enabling the `osa-dispatcher` and `osad` services, and performing initial setup.

OSAD clients use the fully qualified domain name (FQDN) of the server to communicate with the `osa-dispatcher` service.

SSL is required for osad communication. If SSL certificates are not available, the daemon on your client systems will fail to connect. Make sure your firewall rules are set to allow the required ports. For more information, see xref:tab.install.ports.server[Server Ports].

Procedure: Enabling OSAD

1. On your SUSE Manager server, as the root user, start the osa-dispatcher service:

```
systemctl start osa-dispatcher
```

2. On each client machine, install the osad package from the Tools child channel. The osad package should be installed on clients only. If you install the osad package on your SUSE Manager Server, it will conflict with the osa-dispatcher package.
3. On the client systems, as the root user, start the osad service:

```
systemctl start osad
```

   Because osad and osa-dispatcher are run as services, you can use standard commands to manage them, including stop, restart, and status.

Configuration and Log Files

Each OSAD component is configured by local configuration files. We recommend you keep the default configuration parameters for all OSAD components.

Component	Location	Path to Configuration File
osa-dispatcher	Server	/etc/rhn/rhn.conf Section: OSA configuration
osad	Client	/etc/sysconfig/rhn/osad.conf /etc/syseconfig/rhn/up2date
osad log file	Client	/var/log/osad
jabberd log file	Both	/var/log/messages

Troubleshooting OSAD

If your OSAD clients cannot connect to the server, or if the jabberd service takes a lot of time responding to port 5552, it could be because you have exceeded the open file count.

Every client needs one always-open TCP connection to the server, which consumes a single file handler. If the number of file handlers currently open exceeds the maximum number of files that `jabberd` is allowed to use, `jabberd` will queue the requests, and refuse connections.

To resolve this issue, you can increase the file limits for `jabberd` by editing the `/etc/security/limits.conf` configuration file and adding these lines:

```
jabbersoftnofile5100
jabberhardnofile6000
```

Calculate the limits required for your environment by adding 100 to the number of clients for the soft limit, and 1000 to the current number of clients for the soft limit. In the example above, we have assumed 500 current clients, so the soft limit is 5100, and the hard limit is 6000.

You will also need to update the `max_fds` parameter in the `/etc/jabberd/c2s.xml` file with your chosen hard limit:

```
<max_fds>6000</max_fds>
```

Traditional SSH Push

ROOT:partial$entities.adoc

Traditional SSH Pull

ROOT:partial$entities.adoc

Repositories

Repository Types

ROOT:partial$entities.adoc

Description...

Pool Repositories

ROOT:partial$entities.adoc

Devel Repositories

ROOT:partial$entities.adoc

Tool Repositories

ROOT:partial$entities.adoc

Maintenance Repositories

ROOT:partial$entities.adoc

Repository Synchronization

ROOT:partial$entities.adoc

Component Index

Introduction

Description...

Apache

ROOT:partial$entities.adoc

Functions

Apache is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Handles HTTP(S) communication**
- **Serves Static Files**
- **HTTP gateway to: Apache Tomcat, the Python XMLRPC server and Cobbler**

Log Files

Logs for Apache are located in:

```
/var/log/apache2/error_log
```

Apache Tomcat

ROOT:partial$entities.adoc

Functions

Apache Tomcat is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Contains servlet (Java) applications**
- **The most important servlet is the RHN servlet:**
- **Handles the majority of the Web UI**
- **Public XMLRPC API**

Log Files

Logs for Apache Tomcat are located in:

```
/var/log/rhn/rhn_web_ui.log
/var/log/rhn/rhn_web_api.log
/var/log/tomcat/catalina.out
/var/log/tomcat/catalina.*.log
```

Python XMLRPC Server

ROOT:partial$entities.adoc

Functions

The Python XMLRPC Server is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Provides the private XMLRPC API**
- **Used primarily by client tools (mgr_check)**

Log Files

Logs for the Python XMLRPC Server are located in:

```
/var/log/apache2/error_log
/var/log/rhn/rhn_server_xmlrpc.log
```

Taskomatic

ROOT:partial$entities.adoc

Functions

Taskomatic is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Taskomatic handles most background jobs**
- **Patch applicability status refresh**
- **Server side scheduling**
- **SSH push**

- **Cobbler database sync**
- **Repository synchronization and repository metadata generation**
- **CVE audit pre-computation**
- **Cleanup Jobs**

Log Files

Log files for taskomatic are located in:

```
/var/log/rhn/rhn_taskomatic_daemon.log
/var/log/rhn/reposync/*
```

Database

ROOT:partial$entities.adoc

Functions

The database is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Primarily stores application data**
- **Functions as a data exchange area between components**

Log Files

Logs for the database are located in:

```
/var/lib/pgsql/data/pg_log/*
```

mgr-sync

ROOT:partial$entities.adoc

Functions

mgr-sync is a command line tool for SUSE Manager. It performs the following function.

- **mgr-sync is a command line tool that synchronizes with SUSE Customer Center(SCC) and retrieves data and package repositories.**

mgr-sync and Open Source Distributions

This tool is designed for use with a support subscription or trial account with SUSE Customer Center. It is not required for open source distributions(OpenSUSE Leap , CentOS, Ubuntu, etc.).

mgr-sync --help

*The following options are available for the **mgr-sync** command:*

```
mgr-sync --help
usage: mgr-sync [-h] [--version] [-v] [-s] [-d {1,2,3}]
                {list,add,refresh,delete} ...

Synchronize SUSE Manager repositories.

optional arguments:
  -h, --help            show this help message and exit
  --version             Print mgr-sync version
  -v, --verbose         Be verbose
  -s, --store-credentials
                        Store credentials to the local dot file.
  -d {1,2,3}, --debug {1,2,3}
                        Log additional debug information depending on DEBUG

Subcommands:
  {list,add,refresh,delete}
    list                List channels, SCC organization credentials or
                        products
    add                 add channels, SCC organization credentials or products
    refresh             Refresh product, channel and subscription
    delete              Delete SCC organization credentials
```

Log Files

Logs for the mgr-sync tool are located in:

```
/var/log/rhn/mgr-sync.log
/var/log/rhn/rhn_web_api.log
```

spacewalk-repo-sync

ROOT:partial$entities.adoc

Functions

spacewalk-repo-sync is a command line tool for SUSE Manager. It performs the following functions.

- **Copies a repo's metadata to the database**

- **Copies a repo's RPM files to the filesystem**

mgr-sync --help

*The following options are available for the **spacewalk-repo-sync** tool:*

```
spacewalk-repo-sync --help
Usage: spacewalk-repo-sync [options]

Options:
  -h, --help            show this help message and exit
  -l, --list            List the custom channels with the assosiated
                        repositories.
  -s, --show-packages   List all packages in a specified channel.
  -u URL, --url=URL     The url of the repository. Can be used multiple times.
  -c CHANNEL_LABEL, --channel=CHANNEL_LABEL
                        The label of the channel to sync packages to. Can be
                        used multiple times.
  -p PARENT_LABEL, --parent-channel=PARENT_LABEL
                        Synchronize the parent channel and all its child
                        channels.
  -d, --dry-run         Test run. No sync takes place.
  --latest              Sync latest packages only. Use carefully - you might
                        need to fix some dependencies on your own.
  -g CONFIG, --config=CONFIG
                        Configuration file
  -t REPO_TYPE, --type=REPO_TYPE
                        Force type of repository ("yum", "uln" and "deb" are
                        supported)
  -f, --fail            If a package import fails, fail the entire operation
  -n, --non-interactive
                        Do not ask anything, use default answers
  -i FILTERS, --include=FILTERS
                        Comma or space separated list of included packages or
                        package groups.
  -e FILTERS, --exclude=FILTERS
                        Comma or space separated list of excluded packages or
                        package groups.
  --email               e-mail a report of what was synced/imported
  --traceback-mail=TRACEBACK_MAIL
                        alternative email address(es) for sync output (--email
                        option)
  --no-errata           Do not sync errata
  --no-packages         Do not sync packages
  --sync-kickstart      Sync kickstartable tree
  --force-all-errata    Process metadata of all errata, not only missing.
  --batch-size=BATCH_SIZE
                        max. batch size for package import (debug only)
  -Y, --deep-verify     Do not use cached package checksums
  -v, --verbose         Verbose output. Possible to accumulate: -vvv
```

Log Files

*Logs for the **spacewalk-repo-sync** tool are located in:*

```
/var/log/rhn/reposync/*
```

osa-dispatcher

ROOT:partial$entities.adoc

Functions

osa-dispatcher is a component of SUSE Manager. It performs the following function in the stack.

- **Monitors database for actions, informing osad clients when they need to run them**

osa-dispatcher --help

*The following options are available for the **osa-dispatcher**:*

```
osa-dispatcher --help
Usage: osa-dispatcher [options]

Options:
  -v, --verbose        Increase verbosity
  -N, --nodetach       Suppress backgrounding and detachment of the process
  --pid-file=PID_FILE  Write to this PID file
  --logfile=LOGFILE    Write log information to this file
  -h, --help           show this help message and exit
```

Log Files

*Logs for the **osa-dispatcher** are located in:*

```
/var/log/rhn/osa_dispatcher.log
```

jabberd

ROOT:partial$entities.adoc

Functions

jabberd is a component of SUSE Manager. It performs the following function in the stack.

- **Implements the Jabber (XMPP) protocol that osa-dispatcher uses**

Log Files

Logs for jabberd are located in:

```
/var/log/messages
```

mgr_check

ROOT:partial$entities.adoc

Functions

mgr_check is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Client-side command line tool for legacy clients that checks for actions on the server and executes them**

mgr_check and rhn_check

mgr_check is symlinked to **rhn_check** in `/usr/sbin/`. Both *mgr_check* and *rhn_check* can be used for checking for actions on the server.

```
lrwxrwxrwx 1 root root           9 Sep  9 09:05 mgr_check ->
rhn_check*
```

mgr_check --help

*The following options are available for the **rhn_check** on your legacy clients:*

```
mgr_check --help
Usage: rhn_check [options]

Options:
  -v, --verbose         Show additional output. Repeat for more detail.
  --proxy=PROXY         Specify an http proxy to use
  --proxyUser=PROXYUSER
                        Specify a username to use with an authenticated http
                        proxy
  --proxyPassword=PROXYPASSWORD
                        Specify a password to use with an authenticated http
                        proxy
  --version             show program's version number and exit
  -h, --help            show this help message and exit
```

Log Files

*Logs for the **mgr_check** are located on your legacy clients in:*

```
/var/log/up2date
```

zypp-plugin-spacewalk

ROOT:partial$entities.adoc

Functions

zypp-plugin-spacewalk is a component of SUSE Manager. It performs the following functions in the stack.

- **Client-side add-on to zypper for legacy clients**
- **Exposes SUSE Manager channels as zypper repositories**
- **The plugin is not required on salt-minions**

Log Files

*Logs for the **zypp-plugin-spacewalk** are located on your legacy clients in:*

```
/var/log/zypper.log
/var/log/zypp/*
```

rhnsd

ROOT:partial$entities.adoc

Functions

rhnsd is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Client-side daemon for legacy clients**
- **Periodically calls mgr_check(symlinked to rhn_check)**
- **Randomizes check time not to overload the server**

rhnsd --help

The following options are available for use with rhnsd on your legacy clients:

```
rhnsd --help
Usage: rhnsd [OPTION...]
Spacewalk Services Daemon

  -f, --foreground           Run in foreground
  -i, --interval=MINS        Connect to Spacewalk every MINS minutes
  -?, --help                 Give this help list
      --usage                Give a short usage message
  -V, --version              Print program version

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.
```

osad

ROOT:partial$entities.adoc

Functions

osad is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Client-side daemon for legacy clients**
- **Calls mgr_check(rhn_check) when notified by Jabber**

osad --help

The following options are available for use with **osad** on your legacy clients:

```
osad --help
Usage: osad [options]

Options:
  -v, --verbose         Increase verbosity
  -N, --nodetach        Suppress backgrounding and detachment of the process
  --pid-file=PID_FILE   Write to this PID file
  --logfile=LOGFILE     Write log information to this file
  --cfg=CFG             Use this configuration file for defaults
  --jabber-server=JABBER_SERVER
                        Primary jabber server to connect to
  -h, --help            show this help message and exit
```

Log Files

*Logs for **osad** are located in:*

```
/var/log/osad
```

salt-master

ROOT:partial$entities.adoc

Functions

The **salt-master** is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Core process of Salt on the server side**
- **Provides communication with salt minions**
- **Handles Salt jobs, publishes to the Salt event Bus**
- **Handles minion responses**
- **Manages states, highstates, pillar information, etc**

salt-master --help

The following options are available for the **salt-master**. The following list is not comprehensive, for more information see: The Saltstack Docs

Options:

```
salt-master --help
Usage: salt-master [options]

The Salt Master, used to control the Salt Minions

Options:
  --version             show program's version number and exit
  -V, --versions-report
                        Show program's dependencies version number and exit.
  -h, --help            show this help message and exit
  --saltfile=SALTFILE   Specify the path to a Saltfile. If not passed, one
                        will be searched for in the current working directory.
  -c CONFIG_DIR, --config-dir=CONFIG_DIR
                        Pass in an alternative configuration directory.
                        Default: '/etc/salt'.
  -u USER, --user=USER  Specify user to run salt-master.
  -d, --daemon          Run the salt-master as a daemon.
  --pid-file=PIDFILE    Specify the location of the pidfile. Default:
                        '/var/run/salt-master.pid'.

  Logging Options:
    Logging options which override any settings defined on the
    configuration files.

    -l LOG_LEVEL, --log-level=LOG_LEVEL
                        Console logging log level. One of u'all', u'garbage',
                        u'trace', u'debug', u'profile', u'info', u'warning',
                        u'error', u'critical', u'quiet'. Default: 'warning'.
    --log-file=LOG_FILE
                        Log file path. Default: '/var/log/salt/master'.
    --log-file-level=LOG_LEVEL_LOGFILE
                        Logfile logging log level. One of u'all', u'garbage',
                        u'trace', u'debug', u'profile', u'info', u'warning',
                        u'error', u'critical', u'quiet'. Default: 'warning'.

You can find additional help about salt-master issuing "man salt-master" or on
http://docs.saltstack.com
```

Log Files

*Logs for **salt-master** are located in:*

```
/var/log/salt/master
```

salt-api

ROOT:partial$entities.adoc

Functions

The **salt-api** is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Internal API communicates the Java side of SUSE Manager with the salt-master**

- **Provides HTTPS and websocket interfaces with the salt-master**
- **Handles the SSH connections to minions (SSH Push)**

salt-api --help

The following options are available for the **salt-api**. The following list is not comprehensive, for more information see: The Saltstack Docs

Options:

```
salt-api --help
Usage: salt-api [options]

The Salt API system manages network API connectors for the Salt Master

Options:
  --version             show program's version number and exit
  -V, --versions-report
                        Show program's dependencies version number and exit.
  -h, --help            show this help message and exit
  -c CONFIG_DIR, --config-dir=CONFIG_DIR
                        Pass in an alternative configuration directory.
                        Default: '/etc/salt'.
  -d, --daemon          Run the salt-api as a daemon.
  --pid-file=PIDFILE    Specify the location of the pidfile. Default:
                        '/var/run/salt-api.pid'.

  Logging Options:
    Logging options which override any settings defined on the
    configuration files.

    -l LOG_LEVEL, --log-level=LOG_LEVEL
                        Console logging log level. One of u'all', u'garbage',
                        u'trace', u'debug', u'profile', u'info', u'warning',
                        u'error', u'critical', u'quiet'. Default: 'warning'.
    --log-file=API_LOGFILE
                        Log file path. Default: '/var/log/salt/api'.
    --log-file-level=LOG_LEVEL_LOGFILE
                        Logfile logging log level. One of u'all', u'garbage',
                        u'trace', u'debug', u'profile', u'info', u'warning',
                        u'error', u'critical', u'quiet'. Default: 'warning'.

You can find additional help about salt-api issuing "man salt-api" or on
http://docs.saltstack.com
```

Log Files

*Logs for **salt-api** are located in:*

```
/var/log/salt/master
/var/log/salt/api
```

salt-minion

ROOT:partial$entities.adoc

Functions

The **salt-minion** is a primary component of SUSE Manager. It performs the following functions in the stack.

- **Client-side main process for Salt clients (only pull method)**
- **Communicates the client with salt-master via Salt event bus (ZeroMQ)**
- **Executes the actions received from the Salt master on the client (minion)**

salt-minion --help

The following options are available for the **salt-minion**. The following list is not comprehensive, for more information see: The Saltstack Docs

Options:

```
salt-minion --help
Usage: salt-minion [options]

The Salt Minion, receives commands from a remote Salt Master

Options:
  --version             show program's version number and exit
  -V, --versions-report
                        Show program's dependencies version number and exit.
  -h, --help            show this help message and exit
  --saltfile=SALTFILE   Specify the path to a Saltfile. If not passed, one
                        will be searched for in the current working directory.
  -c CONFIG_DIR, --config-dir=CONFIG_DIR
                        Pass in an alternative configuration directory.
                        Default: '/etc/salt'.
  -u USER, --user=USER  Specify user to run salt-minion.
  -d, --daemon          Run the salt-minion as a daemon.
  --pid-file=PIDFILE    Specify the location of the pidfile. Default:
                        '/var/run/salt-minion.pid'.

  Logging Options:
    Logging options which override any settings defined on the
    configuration files.

    -l LOG_LEVEL, --log-level=LOG_LEVEL
                        Console logging log level. One of u'all', u'garbage',
                        u'trace', u'debug', u'profile', u'info', u'warning',
                        u'error', u'critical', u'quiet'. Default: 'warning'.
    --log-file=LOG_FILE
                        Log file path. Default: '/var/log/salt/minion'.
    --log-file-level=LOG_LEVEL_LOGFILE
                        Logfile logging log level. One of u'all', u'garbage',
                        u'trace', u'debug', u'profile', u'info', u'warning',
                        u'error', u'critical', u'quiet'. Default: 'warning'.

You can find additional help about salt-minion issuing "man salt-minion" or on
http://docs.saltstack.com
```

Log Files

*Logs for **salt-minion** are located in:*

```
/var/log/salt/minion
```

salt-broker

ROOT:partial$entities.adoc

Functions

The **salt-broker** is a component of the SUSE Manager proxy. It performs the following functions in the stack.

- **Used only in the SUSE Manager Proxy for minions using pull method**

- **Forwards the ZeroMQ Salt channels from SUSE Manager server to the proxy minions**

Log Files

*Logs for **salt-broker** are located in:*

```
/var/log/salt/broker
```