Cobbler Documentation

Release 3.0.1

Enno Gotthold

May 27, 2020

Contents

cobbler is a provisioning (installation) and update server. It supports deployments via PXE (network booting), virtualization (Xen, QEMU/KVM, or VMware), and re-installs of existing Linux systems. The latter two features are enabled by usage of 'koan' on the remote system. Update server features include yum mirroring and integration of those mirrors with automated installation files. Cobbler has a command line interface, Web UI, and extensive Python and XMLRPC APIs for integration with external scripts and applications.

If you want to explore tools or scripts which are using cobbler please use the Github-Topic: https://github.com/topics/cobbler

Here you should find a comprehensive overview about the usage of cobbler.

CHAPTER 1

Quickstart

Cobbler can be a somewhat complex system to get started with, due to the wide variety of technologies it is designed to manage, but it does support a great deal of functionality immediately after installation with little to no customization needed. Before getting started with Cobbler, you should have a good working knowledge of PXE as well as the automated installation methodology of your chosen distribution(s).

We will assume you have successfully installed Cobbler, please refer to the *Installation Guide* for instructions for your specific operating system. Finally, this part guide will focus only on the CLI application.

1.1 Preparing your OS

1.1.1 SELinux

Before getting started with Cobbler, it may be convenient to either disable SELinux or set it to "permissive" mode, especially if you are unfamiliar with SELinux troubleshooting or modifying SELinux policy. Cobbler constantly evolves to assist in managing new system technologies, and the policy that ships with your OS can sometimes lag behind the feature-set we provide, resulting in AVC denials that break Cobbler's functionality.

1.1.2 Firewall

TBD

1.2 Changing settings

Before starting the cobblerd service, there are a few things you should modify.

Settings are stored in `/etc/cobbler/settings`. This file is a YAML formatted data file, so be sure to take care when editing this file as an incorrectly formatted file will prevent cobblerd from running.

1.2.1 Default encrypted password

This setting controls the root password that is set for new systems during the handsoff installation.

```
default_password_crypted: "$1$bfI7WLZz$PxXetL97LkScqJFxnW7KS1"
```

You should modify this by running the following command and inserting the output into the above string (be sure to save the quote marks):

```
$ openssl passwd -1
```

1.2.2 Server and next_server

The `server` option sets the IP that will be used for the address of the cobbler server. **DO NOT** use 0.0.0.0, as it is not the listening address. This should be set to the IP you want hosts that are being built to contact the Cobbler server on for such protocols as HTTP and TFTP.

```
server: 127.0.0.1
```

The `next_server` option is used for DHCP/PXE as the IP of the TFTP server from which network boot files are downloaded. Usually, this will be the same IP as the server setting.

```
next_server: 127.0.0.1
```

1.3 DHCP management and DHCP server template

In order to PXE boot, you need a DHCP server to hand out addresses and direct the booting system to the TFTP server where it can download the network boot files. Cobbler can manage this for you, via the `manage_dhcp` setting:

```
manage_dhcp: 0
```

Change that setting to 1 so Cobbler will generate the `dhcpd.conf` file based on the `dhcp.template` that is included with Cobbler. This template will most likely need to be modified as well, based on your network settings:

```
$ vi /etc/cobbler/dhcp.template
```

For most uses, you'll only need to modify this block:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    option routers             192.168.1.1;
    option domain-name-servers 192.168.1.210,192.168.1.211;
    option subnet-mask         255.255.255.0;
    filename                   "/pxelinux.0";
    default-lease-time         21600;
    max-lease-time             43200;
    next-server                $next_server;
}
```

No matter what, make sure you do not modify the `next-server $next_server;` line, as that is how the next_server setting is pulled into the configuration. This file is a cheetah template, so be sure not to modify anything starting after this line:

```
#for dhcp_tag in $dhcp_tags.keys():
```

Completely going through the `dhcpd.conf` configuration syntax is beyond the scope of this document, but for more information see the man page for more details:

```
$ man dhcpd.conf
```

1.4 Notes on files and directories

Cobbler makes heavy use of the `/var` directory. The `/var/www/cobbler/distro_mirror` directory is where all of the distrubtion and repository files are copied, so you will need 5-10GB of free space per distribution you wish to import.

If you have installed Cobbler onto a system that has very little free space in the partition containing `/var`, please read the *Relocating your installation* section of the Installation Guide to learn how you can relocate your installation properly.

1.5 Starting and enabling the Cobbler service

Once you have updated your settings, you're ready to start the service:

```
$ systemctl start cobblerd.service
$ systemctl enable cobblerd.service
$ systemctl status cobblerd.service
```

If everything has gone well, you should see output from the status command like this:

```
cobblerd.service - Cobbler Helper Daemon
    Loaded: loaded (/lib/systemd/system/cobblerd.service; enabled)
      Active: active (running) since Sun, 17 Jun 2012 13:01:28 -0500; 1min 44s ago
    Main PID: 1234 (cobblerd)
      CGroup: name=systemd:/system/cobblerd.service
              └ 1234 /usr/bin/python /usr/bin/cobblerd -F
```

1.6 Checking for problems and your first sync

Now that the cobblerd service is up and running, it's time to check for problems. Cobbler's check command will make some suggestions, but it is important to remember that these are mainly only suggestions and probably aren't critical for basic functionality. If you are running iptables or SELinux, it is important to review any messages concering those that check may report.

```
$ cobbler check
The following are potential configuration items that you may want to fix:

1. ....
2. ....
```

Restart cobblerd and then run `cobbler sync` to apply changes.

If you decide to follow any of the suggestions, such as installing extra packages, making configuration changes, etc., be sure to restart the cobblerd service as it suggests so the changes are applied.

Once you are done reviewing the output of `cobbler check`, it is time to synchronize things for the first time. This is not critical, but a failure to properly sync at this point can reveal a configuration problem.

```
$ cobbler sync
task started: 2012-06-24_224243_sync
task started (id=Sync, time=Sun Jun 24 22:42:43 2012)
running pre-sync triggers
...
rendering DHCP files
generating /etc/dhcp/dhcpd.conf
cleaning link caches
running: find /var/lib/tftpboot/images/.link_cache -maxdepth 1 -type f -links 1 -
↪exec rm -f '{}' ';'
```

(continues on next page)

(continued from previous page)

```
received on stdout:
received on stderr:
running post-sync triggers
running python triggers from /var/lib/cobbler/triggers/sync/post/*
running python trigger cobbler.modules.sync_post_restart_services
running: dhcpd -t -q
received on stdout:
received on stderr:
running: service dhcpd restart
received on stdout:
received on stderr:
running shell triggers from /var/lib/cobbler/triggers/sync/post/*
running python triggers from /var/lib/cobbler/triggers/change/*
running python trigger cobbler.modules.scm_track
running shell triggers from /var/lib/cobbler/triggers/change/*
*** TASK COMPLETE ***
```

Assuming all went well and no errors were reported, you are ready to move on to the next step.

1.7 Importing your first distribution

Cobbler automates adding distributions and profiles via the `cobbler import` command. This command can (usually) automatically detect the type and version of the distribution your importing and create (one or more) profiles with the correct settings for you.

1.7.1 Download an ISO image

In order to import a distribution, you will need a DVD ISO for your distribution.

NOTE: You must use a full DVD, and not a "Live CD" ISO. For this example, we'll be using the Fedora 17 x86_64 ISO.

Once this file is downloaded, mount it somewhere:

```
$ mount -t iso9660 -o loop,ro /path/to/isos/Fedora-17-x86_64-DVD.iso /mnt
```

1.7.2 Run the import

You are now ready to import the distribution. The name and path arguments are the only required options for import:

```
$ cobbler import --name=fedora17 --arch=x86_64 --path=/mnt
```

The `--arch` option need not be specified, as it will normally be auto-detected. We're doing so in this example in order to prevent multiple architectures from being found.

Listing objects

If no errors were reported during the import, you can view details about the distros and profiles that were created during the import.

```
$ cobbler distro list
$ cobbler profile list
```

The import command will typically create at least one distro/profile pair, which will have the same name as shown above. In some cases (for instance when a xen-based kernel is found), more than one distro/profile pair will be created.

Object details

The report command shows the details of objects in cobbler:

```
$ cobbler distro report --name=fedora17-x86_64
Name                           : fedora17-x86_64
Architecture                   : x86_64
TFTP Boot Files                : {}
Breed                          : redhat
Comment                        :
Fetchable Files                : {}
Initrd                         : /var/www/cobbler/distro_mirror/fedora17-x86_64/
↪images/pxeboot/initrd.img
Kernel                         : /var/www/cobbler/distro_mirror/fedora17-x86_64/
↪images/pxeboot/vmlinuz
Kernel Options                 : {}
Kernel Options (Post Install)  : {}
Automatic Installation Template Metadata : {'tree': 'http://@@http_server@@/cblr/
↪links/fedora17-x86_64'}
Management Classes             : []
OS Version                     : fedora17
Owners                         : ['admin']
Red Hat Management Key         : <<inherit>>
Red Hat Management Server      : <<inherit>>
Template Files                 : {}
```

As you can see above, the import command filled out quite a few fields automatically, such as the breed, OS version, and initrd/kernel file locations. The "Automatic Installation Template Metadata" field (`--autoinstall_meta` internally) is used for miscellaneous variables, and contains the critical "tree" variable. This is used in the automated installation templates to specify the URL where the installation files can be found.

Something else to note: some fields are set to `<<inherit>>`. This means they will use either the default setting (found in the settings file), or (in the case of profiles, sub-profiles, and systems) will use whatever is set in the parent object.

Creating a system

Now that you have a distro and profile, you can create a system. Profiles can be used to PXE boot, but most of the features in cobbler revolve around system objects. The more information you give about a system, the more cobbler will do automatically for you.

First, we'll create a system object based on the profile that was created during the import. When creating a system, the name and profile are the only two required fields:

```
$ cobbler system add --name=test --profile=fedora17-x86_64
$ cobbler system list
test
$ cobbler system report --name=test
Name                           : test
TFTP Boot Files                : {}
Comment                        :
Enable gPXE?                   : 0
Fetchable Files                : {}
Gateway                        :
Hostname                       :
```

(continues on next page)

(continued from previous page)

```
Image                          :
IPv6 Autoconfiguration         : False
IPv6 Default Device            :
Kernel Options                 : {}
Kernel Options (Post Install)  : {}
Automatic Installation Template: <<inherit>>
Automatic Installation Template Metadata: {}
Management Classes             : []
Management Parameters          : <<inherit>>
Name Servers                   : []
Name Servers Search Path       : []
Netboot Enabled                : True
Owners                         : ['admin']
Power Management Address       :
Power Management ID            :
Power Management Password      :
Power Management Type          : ipmitool
Power Management Username      :
Profile                        : fedora17-x86_64
Proxy                          : <<inherit>>
Red Hat Management Key         : <<inherit>>
Red Hat Management Server      : <<inherit>>
Repos Enabled                  : False
Server Override                : <<inherit>>
Status                         : production
Template Files                 : {}
Virt Auto Boot                 : <<inherit>>
Virt CPUs                      : <<inherit>>
Virt Disk Driver Type          : <<inherit>>
Virt File Size(GB)             : <<inherit>>
Virt Path                      : <<inherit>>
Virt RAM (MB)                  : <<inherit>>
Virt Type                      : <<inherit>>
```

The primary reason for creating a system object is network configuration. When using profiles, you're limited to DHCP interfaces, but with systems you can specify many more network configuration options.

So now we'll setup a single, simple interface in the `192.168.1/24` network:

```
$ cobbler system edit --name=test --interface=eth0 --mac=00:11:22:AA:BB:CC --ip-
↪address=192.168.1.100 --netmask=255.255.255.0 --static=1 --dns-name=test.
↪mydomain.com
```

The default gateway isn't specified per-NIC, so just add that separately (along with the hostname):

```
$ cobbler system edit --name=test --gateway=192.168.1.1 --hostname=test.mydomain.
↪com
```

The `--hostname` field corresponds to the local system name and is returned by the `hostname` command. The `--dns-name` (which can be set per-NIC) should correspond to a DNS A-record tied to the IP of that interface. Neither are required, but it is a good practice to specify both. Some advanced features (like configuration management) rely on the `--dns-name` field for system record look-ups.

Whenever a system is edited, cobbler executes what is known as a "lite sync", which regenerates critical files like the PXE boot file in the TFTP root directory. One thing it will **NOT** do is execute service management actions, like regenerating the `dhcpd.conf` and restarting the DHCP service. After adding a system with a static interface it is a good idea to execute a full `cobbler sync` to ensure the dhcpd.conf file is rewritten with the correct static lease and the service is bounced.

CHAPTER 2

Install Guide

Setting up and running cobblerd is not a easy task. Knowledge in apache configuration (setting up ssl, virtual hosts, apache module and wsgi) is needed. Certificates and some server administration knowledge is required too.

Cobbler is available for installation in several different ways, through packaging systems for each distribution or directly from source.

Cobbler has both definite and optional prerequisites, based on the features you'd like to use. This section documents the definite prerequisites for both a basic installation and when building/installing from source.

2.1 Prerequisites

2.1.1 Packages

Please note that installing any of the packages here via a package manager (such as dnf/yum or apt) can and will require a large number of ancilary packages, which we do not document here. The package definition should automatically pull these packages in and install them along with Cobbler, however it is always best to verify these requirements have been met prior to installing cobbler or any of its components.

First and foremost, Cobbler requires Python. Any 2.x version should work for 2.8.x releases. Since 3.0.0 you will need Python 3. Cobbler also requires the installation of the following packages:

- createrepo
- httpd / apache2
- xorriso
- mod_wsgi / libapache2-mod-wsgi
- mod_ssl / libapache2-mod-ssl
- python-cheetah
- python-netaddr
- python-simplejson
- PyYAML / python-yaml
- rsync

- syslinux
- tftp-server / atftpd
- yum-utils

Cobbler-web only has one other requirement besides Cobbler itself:

- Django / python-django

Koan can be installed apart from Cobbler, and has only the following requirement (besides python itself of course):

- python-simplejson

2.1.2 Source

Installation from source requires the following additional software:

- git
- make
- python-devel
- python-cheetah
- openssl

2.2 Installation

Cobbler is available for installation for many Linux variants through their native packaging systems. However, the Cobbler project also provides packages for all supported distributions which is the preferred method of installation.

2.2.1 Packages

We leave packaging to downstream; this means you have to check the repositories provided by your distribution vendor. However we provide docker files for

- CentOS 7
- CentOS 8
- Debian 10 Buster

which will give you packages which will work better then building from source yourself.

2.2.2 Packages from source

For some platforms it's also possible to build packages directly from the source tree.

2.3 RPM

```
$ make rpms
... (lots of output) ...
Wrote: /path/to/cobbler/rpm-build/cobbler-3.0.0-1.fc20.src.rpm
Wrote: /path/to/cobbler/rpm-build/cobbler-3.0.0-1.fc20.noarch.rpm
Wrote: /path/to/cobbler/rpm-build/koan-3.0.0-1.fc20.noarch.rpm
Wrote: /path/to/cobbler/rpm-build/cobbler-web-3.0.0-1.fc20.noarch.rpm
```

As you can see, an RPM is output for each component of Cobbler, as well as a source RPM. This command was run on a system running Fedora 20, hence the fc20 in the RPM name - this will be different based on the distribution you're running.

2.4 DEB

To install Cobbler from source on a Debian-Based system, the following steps need to be made (tested on Debian Buster):

```
$ apt-get -y install make git
$ apt-get -y install python3-yaml python3-cheetah python3-netaddr python3-
↪simplejson
$ apt-get -y install python3-future python3-distro python3-setuptools python3-
↪sphinx python3-coverage
$ apt-get -y install pyflakes3 python3-pycodestyle
$ apt-get -y install apache2 libapache2-mod-wsgi-py3
$ apt-get -y install atftpd
# In case you want cobbler-web
$ apt-get -y install python3-django

$ a2enmod proxy
$ a2enmod proxy_http
$ a2enmod rewrite

$ ln -s /srv/tftp /var/lib/tftpboot

$ systemctl restart apache2
```

Change all `/var/www/cobbler` in `/etc/apache2/conf.d/cobbler.conf` to `/usr/share/cobbler/webroot/` Init script: - add Required-Stop line - path needs to be `/usr/local/...` or fix the install location

2.4.1 Source

The latest source code is available through git:

```
$ git clone https://github.com/cobbler/cobbler.git
$ cd cobbler
```

The release30 branch corresponds to the official release version for the 3.0.x series. The master branch is the development series, and always uses an odd number for the minor version (for example, 3.1.0).

When building from source, make sure you have the correct prerequisites. Once they are, you can install Cobbler with the following command:

```
$ make install
```

This command will rewrite all configuration files on your system if you have an existing installation of Cobbler (whether it was installed via packages or from an older source tree). To preserve your existing configuration files, snippets and automatic installation files, run this command:

```
$ make devinstall
```

To install the Cobbler web GUI, use this command:

```
$ make webtest
```

This will do a full install, not just the web GUI. `make webtest` is a wrapper around `make devinstall`, so your configuration files will also be saved when running this command. Be adviced that we don't copy the

service file into the correct directory and that the path to the binary may be wrong depending on the location of the binary on your system. Do this manually and then you should be good to go. The same is valid for the Apache2 webserver config.

Also note that this is not enough to run Cobbler-Web. Cobbler web needs the directories `/usr/share/cobbler/web` with the file `cobbler.wsgi` in it. This is currently a manual step. Also remember to manually enter a value for `SECRET_KEY` in `settings.py` and copy that to above mentioned directory as well as the templates directory.

2.5 Relocating your installation

Often folks don't have a very large `/var` partition, which is what Cobbler uses by default for mirroring install trees and the like.

You'll notice you can reconfigure the webdir location just by going into `/etc/cobbler/settings`, but it's not the best way to do things – especially as the packaging process does include some files and directories in the stock path. This means that, for upgrades and the like, you'll be breaking things somewhat. Rather than attempting to reconfigure Cobbler, your Apache configuration, your file permissions, and your SELinux rules, the recommended course of action is very simple.

1. Copy everything you have already in `/var/www/cobbler` to another location – for instance, `/opt/cobbler_data`
2. Now just create a symlink or bind mount at `/var/www/cobbler` that points to `/opt/cobbler_data`.

Done. You're up and running.

If you decided to access Cobbler's data store over NFS (not recommended) you really want to mount NFS on `/var/www/cobbler` with SELinux context passed in as a parameter to mount versus the symlink. You may also have to deal with problems related to rootsquash. However if you are making a mirror of a Cobbler server for a multi-site setup, mounting read only is ok there.

Also Note: `/var/lib/cobbler` can not live on NFS, as this interferes with locking ("flock") Cobbler does around it's storage files.

CHAPTER 3

Cobbler CLI

This page contains a description for commands which can be used from the CLI.

3.1 General Principles

This should just be a brief overview. For the detailed expanations please reefer to Readthedocs.

3.1.1 Distros, Profiles and Systems

Cobbler has a system of inheritance when it comes to managing the information you want to apply to a certain system.

3.1.2 Images

3.1.3 Repositorys

3.1.4 Management Classes

3.1.5 Deleting configuration entries

If you want to remove a specific object, use the remove command with the name that was used to add it.

```
cobbler distro|profile|system|repo|image|mgmtclass|package|file remove --
↪name=string
```

3.1.6 Editing

If you want to change a particular setting without doing an `add` again, use the `edit` command, using the same name you gave when you added the item. Anything supplied in the parameter list will overwrite the settings in the existing object, preserving settings not mentioned.

```
cobbler distro|profile|system|repo|image|mgmtclass|package|file edit --name=string␣
↪[parameterlist]
```

3.1.7 Copying

Objects can also be copied:

```
cobbler distro|profile|system|repo|image|mgmtclass|package|file copy --
↪name=oldname --newname=newname
```

3.1.8 Renaming

Objects can also be renamed, as long as other objects don't reference them.

```
cobbler distro|profile|system|repo|image|mgmtclass|package|file rename --
↪name=oldname --newname=newname
```

3.2 CLI-Commands

Short Usage: `cobbler command [subcommand] [--arg1=value1] [--arg2=value2]`

Long Usage:

```
cobbler <distro|profile|system|repo|image|mgmtclass|package|file> ...␣
↪[add|edit|copy|get-autoinstall*|list|remove|rename|report] [options|--help]
cobbler <aclsetup|buildiso|import|list|replicate|report|reposync|sync|validate-
↪autoinstalls|version|signature|get-loaders|hardlink> [options|--help]
```

3.2.1 cobbler distro

This first step towards configuring what you want to install is to add a distribution record to cobbler's configuration.

If there is an rsync mirror, DVD, NFS, or filesystem tree available that you would rather `import` instead, skip down to the documentation about the `import` command. It's really a lot easier to follow the import workflow – it only requires waiting for the mirror content to be copied and/or scanned. Imported mirrors also save time during install since they don't have to hit external install sources.

If you want to be explicit with distribution definition, however, here's how it works:

```
$ cobbler distro add --name=string --kernel=path --initrd=path [--kopts=string] [--
↪kopts-post=string] [--ksmeta=string] [--arch=i386|x86_64|ppc|ppc64] [--
↪breed=redhat|debian|suse] [--template-files=string]
```

Name	Description
name	a string identifying the distribution, this should be something like `rhel6`.
kernel	An absolute filesystem path to a kernel image.
initrd	An absolute filesystem path to a initrd image.
remote-boot-kernel	A URL pointing to the installation initrd of a distribution. If the bootloader has this support, it will directly download the kernel from this URL, instead of the directory of the tftp client. Note: The kernel (or initrd below) will still be copied into the image directory of the tftp server. The above kernel parameter is still needed (e.g. to build iso images, etc.). The advantage of letting the boot loader retrieve the kernel/initrd directly is the support of changing/updated distibutions. E.g. openSUSE Tumbleweed is updated on the fly and if cobbler would copy/cache the kernel/initrd in the tftp directory, you would get a "kernel does not match distribution" (or similar) error when trying to install.
remote-boot-initrd	See remote-boot-kernel above.
kopts	Sets kernel command-line arguments that the distro, and profiles/systems depending on it, will use. To remove a kernel argument that may be added by a higher cobbler object (or in the global settings), you can prefix it with a `!`.
	Example: `--kopts="foo=bar baz=3 asdf !gulp"`
	This example passes the arguments `foo=bar baz=3 asdf` but will make sure `gulp` is not passed even if it was requested at a level higher up in the cobbler configuration.
kopts-post	This is just like `--kopts`, though it governs kernel options on the installed OS, as opposed to kernel options fed to the installer. The syntax is exactly the same. This requires some special snippets to be found in your automatic installation template in order for this to work. Automatic installation templating is described later on in this document.
	Example: `noapic`
arch	Sets the architecture for the PXE bootloader and also controls how koan's `--replace-self` option will operate.
	The default setting (`standard`) will use `pxelinux`. Set to `ppc` and `ppc64` to use `yaboot`.
	`x86` and `x86_64` effectively do the same thing as standard.
	If you perform a `cobbler import`, the arch field will be auto-assigned.
ksmeta	This is an advanced feature that sets automatic installation template variables to substitute, thus enabling those files to be treated as templates. Templates are powered using Cheetah and are described further along in this manpage as well as on the Cobbler Wiki.
	Example: `--ksmeta="foo=bar baz=3 asdf"`
	See the section on "Kickstart Templating" for further information.
breed	Controls how various physical and virtual parameters, including kernel arguments for automatic installation, are to be treated. Defaults to `redhat`, which is a suitable value for Fedora and CentOS as well. It means anything redhat based.
	There is limited experimental support for specifying "debian", "ubuntu", or "suse", which treats the automatic installation template file as a preseed/autoyast file format and changes the kernel arguments appropriately. Support for other types of distributions is possible in the future. See the Wiki for the latest information about support for these distributions.
	The file used for the answer file, regardless of the breed setting, is the value used for `--autoinst` when creating the profile.
os-version	Generally this field can be ignored. It is intended to alter some hardware setup for virtualized instances when provisioning guests with koan. The valid options for `--os-version` vary depending on what is specified for `--breed`. If you specify an invalid option, the error message will contain a list of valid os versions that can be used. If you don't know the os version or it does not appear in the list, omitting this argument or using `other` should be perfectly fine. If you don't encounter any problems with virtualized instances, this option can be safely ignored.
owners	Users with small sites and a limited number of admins can probably ignore this option. All cobbler objects (distros, profiles, systems, and repos) can take a –owners parameter to specify what cobbler users can edit particular objects.This only applies to the Cobbler WebUI and XMLRPC interface, not the "cobbler" command line tool run from the shell. Furthermore, this is only respected by the `authz_ownership` module which must be enabled in `/etc/cobbler/modules.conf`. The value for `--owners` is a space separated list of users and groups as specified in `/etc/cobbler/users.conf`. For more information see the users.conf file as well as the Cobbler Wiki. In the default Cobbler configuration, this value is completely ignored, as is `users.conf`.
template-files	This feature allows cobbler to be used as a configuration management system. The argument is a space delimited string of `key=value` pairs. Each key is the path to a template file, each value is the path to

3.2.2 cobbler profile

A profile associates a distribution to additional specialized options, such as a installation automation file. Profiles are the core unit of provisioning and at least one profile must exist for every distribution to be provisioned. A profile might represent, for instance, a web server or desktop configuration. In this way, profiles define a role to be performed.

```
$ cobbler profile add --name=string --distro=string [--autoinst=path] [--
↪kopts=string] [--ksmeta=string] [--name-servers=string] [--name-servers-
↪search=string] [--virt-file-size=gigabytes] [--virt-ram=megabytes] [--virt-
↪type=string] [--virt-cpus=integer] [--virt-path=string] [--virt-bridge=string] [-
↪-server] [--parent=profile] [--filename=string]
```

Arguments are the same as listed for distributions, save for the removal of “arch” and “breed”, and with the additions listed below:

Name	Description
name	A descriptive name. This could be something like `rhel5webservers` or `f9desktops`.
distro	The name of a previously defined cobbler distribution. This value is required.
autoinst	Local filesystem path to a automatic installation file, the file must reside under `/var/lib/cobbler/autoinstall_templates`
name-servers	If your nameservers are not provided by DHCP, you can specify a space separated list of addresses here to configure each of the installed nodes to use them (provided the automatic installation files used are installed on a per-system basis). Users with DHCP setups should not need to use this option. This is available to set in profiles to avoid having to set it repeatedly for each system record.
name-servers-search	You can specify a space separated list of domain names to configure each of the installed nodes to use them as domain search path. This is available to set in profiles to avoid having to set it repeatedly for each system record.
virt-file-size	(Virt-only) How large the disk image should be in Gigabytes. The default is 5. This can be a comma separated list (ex: `5,6,7`) to allow for multiple disks of different sizes depending on what is given to `--virt-path`. This should be input as a integer or decimal value without units.
virt-ram	(Virt-only) How many megabytes of RAM to consume. The default is 512 MB. This should be input as an integer without units.
virt-type	(Virt-only) Koan can install images using either Xen paravirt (`xenpv`) or QEMU/KVM (`qemu`). Choose one or the other strings to specify, or values will default to attempting to find a compatible installation type on the client system("auto"). See the "koan" manpage for more documentation. The default `--virt-type` can be configured in the cobbler settings file such that this parameter does not have to be provided. Other virtualization types are supported, for information on those options (such as VMware), see the Cobbler Wiki.
virt-cpus	(Virt-only) How many virtual CPUs should koan give the virtual machine? The default is 1. This is an integer.
virt-path	(Virt-only) Where to store the virtual image on the host system. Except for advanced cases, this parameter can usually be omitted. For disk images, the value is usually an absolute path to an existing directory with an optional filename component. There is support for specifying partitions `/dev/sda4` or volume groups `VolGroup00`, etc.
	For multiple disks, separate the values with commas such as `VolGroup00,VolGroup00` or `/dev/sda4,/dev/sda5`. Both those examples would create two disks for the VM.
virt-bridge	(Virt-only) This specifies the default bridge to use for all systems defined under this profile. If not specified, itwill assume the default value in the cobbler settings file, which as shipped in the RPM is `xenbr0`.
	If using KVM, this is most likely not correct. You may want to override this setting in the system object. Bridge settings are important as they define how outside networking will reach the guest. For more infor-

3.2.3 cobbler system

System records map a piece of hardware (or a virtual machine) with the cobbler profile to be assigned to run on it. This may be thought of as choosing a role for a specific system.

Note that if provisioning via koan and PXE menus alone, it is not required to create system records in cobbler, though they are useful when system specific customizations are required. One such customization would be defining the MAC address. If there is a specific role intended for a given machine, system records should be created for it.

System commands have a wider variety of control offered over network details. In order to use these to the fullest possible extent, the automatic installation template used by cobbler must contain certain automatic installation snippets (sections of code specifically written for Cobbler to make these values become reality). Compare your automatic installation templates with the stock ones in /var/lib/cobbler/autoinstall_templates if you have upgraded, to make sure you can take advantage of all options to their fullest potential. If you are a new cobbler user, base your automatic installation templates off of these templates.

Read more about networking setup at: https://github.com/cobbler/cobbler/wiki/Advanced-networking

Example:

```
$ cobbler system add --name=string --profile=string [--mac=macaddress] [--ip-
↪address=ipaddress] [--hostname=hostname] [--kopts=string] [--ksmeta=string] [--
↪autoinst=path] [--netboot-enabled=Y/N] [--server=string] [--gateway=string] [--
↪dns-name=string] [--static-routes=string] [--power-address=string] [--power-
↪type=string] [--power-user=string] [--power-pass=string] [--power-id=string]
```

Adds a cobbler System to the configuration. Arguments are specified as per "profile add" with the following changes:

Name	Description
name	The system name works like the name option for other command
	If the name looks like a MAC address or an IP, the name will imp
	A system created with name "default" has special semantics. If a
	When using "default" name, don't specify any other arguments t
mac	Specifying a mac address via –mac allows the system object to b
	MAC addresses have the format AA:BB:CC:DD:EE:FF. It's high
	Cobbler does contain a feature (enabled in /etc/cobbler/settings)
ip-address	If cobbler is configured to generate a DHCP configuration (see a
	Example: –ip-address=192.168.1.50
	If DHCP management is disabled and the interface is labelled –s
	Special feature: To control the default PXE behavior for an entir
	When using the CIDR notation trick, don't specify any argument
dns-name	If using the DNS management feature (see advanced section – co
	Example: –dns-name=mycomputer.example.com
	This is a per-interface parameter. If you have multiple interfaces,
gateway and netmask	If you are using static IP configurations and the interface is flagg
	Netmask is a per-interface parameter. Because of the way gatewa
if-gateway	If you are using static IP configurations and have multiple interfa
	This is a per-interface setting.
hostname	This field corresponds to the hostname set in a systems /etc/sysco
	This parameter is assigned once per system, it is not a per-interfa
power-address, power-type, power-user, power-pass, power-id	Cobbler contains features that enable integration with power man
static	Indicates that this interface is statically configured. Many fields (
	This is a per-interface setting.
static-routes	This is a space delimited list of ip/mask:gateway routing informa
	This is a per-interface setting.
virt-bridge	(Virt-only) While –virt-bridge is present in the profile object (see

Name	Description
	This is a per-interface setting.
autoinst	While it is recommended that the –autoinst parameter is only use
netboot-enabled	If set false, the system will be provisionable through koan but no
repos-enabled	If set true, koan can reconfigure repositories after installation. Th
dhcp-tag	If you are setting up a PXE environment with multiple subnets/g
	By default, the dhcp tag for all systems is “default” and means th
	This is described further on the Cobbler Wiki.
interface	By default flags like –ip, –mac, –dhcp-tag, –dns-name, –netmask
	Interface naming notes:
	Additional interfaces can be specified (for example: eth1, or any
	Example:
	cobbler system edit –name=foo –ip-address=192.168.1.50 –mac=
	cobbler system edit –name=foo –interface=eth0 –ip-address=192
	cobbler system report foo
	Interfaces can be deleted using the –delete-interface option.
	Example:
	cobbler system edit –name=foo –interface=eth2 –delete-interface
interface-type, interface-master and bonding-opts/bridge-opts	One of the other advanced networking features supported by Cob
	Example:
	cobbler system edit –name=foo –interface=eth0 –mac=AA:BB:C
	cobbler system edit –name=foo –interface=eth1 –mac=AA:BB:C
	cobbler system edit –name=foo –interface=bond0 –interface-type
	More information about networking setup is available at https://g
	To review what networking configuration you have for any objec
	Example:
	cobbler system report –name=foo

3.2.4 cobbler repo

Repository mirroring allows cobbler to mirror not only install trees (“cobbler import” does this for you) but also optional packages, 3rd party content, and even updates. Mirroring all of this content locally on your network will result in faster, more up-to-date installations and faster updates. If you are only provisioning a home setup, this will probably be overkill, though it can be very useful for larger setups (labs, datacenters, etc).

```
$ cobbler repo add --mirror=url --name=string [--rpmlist=list] [--creatrepo-
↪flags=string] [--keep-updated=Y/N] [--priority=number] [--arch=string] [--mirror-
↪locally=Y/N] [--breed=yum|rsync|rhn]
```

Name	Description
mirror	The address of the yum mirror. This can be an `rsync://`-URL, an ssh location, or a `http://` or `ftp://` mirror location. Filesystem paths also work.
	The mirror address should specify an exact repository to mirror – just one architecture and just one distribution. If you have a separate repo to mirror for a different arch, add that repo separately.
	Here's an example of what looks like a good URL:
	• `rsync://yourmirror.example.com/fedora-linux-core/updates/6/i386` (for rsync protocol) • `http://mirrors.kernel.org/fedora/extras/6/i386/` (for http) • `user@yourmirror.example.com/fedora-linux-core/updates/6/i386` (for SSH)
	Experimental support is also provided for mirroring RHN content when you need a fast local mirror. The mirror syntax for this is `--mirror=rhn://channel-name` and you must have entitlements for this to work. This requires the cobbler server to be installed on RHEL5 or later. You will also need a version of `yum-utils` equal or greater to 1.0.4.
name	This name is used as the save location for the mirror. If the mirror represented, say, Fedora Core 6 i386 updates, a good name would be `fc6i386updates`. Again, be specific.
	This name corresponds with values given to the `--repos` parameter of `cobbler profile add`. If a profile has a `--repos`-value that matches the name given here, that repo can be automatically set up during provisioning (when supported) and installed systems will also use the boot server as a mirror (unless `yum_post_install_mirror` is disabled in the settings file). By default the provisioning server will act as a mirror to systems it installs, which may not be desirable for laptop configurations, etc.
	Distros that can make use of yum repositories during automatic installation include FC6 and later, RHEL 5 and later, and derivative distributions.
	See the documentation on `cobbler profile add` for more information.
rpm-list	By specifying a space-delimited list of package names for `--rpm-list`, one can decide to mirror only a part of a repo (the list of packages given, plus dependencies). This may be helpful in conserving time/space/bandwidth. For instance, when mirroring FC6 Extras, it may be desired to mirror just cobbler and koan, and skip all of the game packages. To do this, use `--rpm-list="cobbler koan"`.
	This option only works for `http://` and `ftp://` repositories (as it is powered by yumdownloader). It will be ignored for other mirror types, such as local paths and `rsync://` mirrors.
createrepo-flags	Specifies optional flags to feed into the createrepo tool, which is called when `cobbler reposync` is run for the given repository. The defaults are `-c cache`.
keep-updated	Specifies that the named repository should not be updated during a normal "cobbler reposync". The repo may still beupdated by name. The repo should be

3.2.5 cobbler image

Example:

```
$ cobbler image
```

3.2.6 cobbler mgmtclass

Management classes allows cobbler to function as an configuration management system. Cobbler currently supports the following resource types:

1. Packages
2. Files

Resources are executed in the order listed above.

```
$ cobbler mgmtclass add --name=string --comment=string [--packages=list] [--
↪files=list]
```

Name	Description
name	The name of the mgmtclass. Use this name when adding a management class to a system, profile, or distro. To add amgmtclass to an existing system use something like (`cobbler system edit --name="madhatter" --mgmt-classes="http mysql"`).
comment	A comment that describes the functions of the management class.
packages	Specifies a list of package resources required by the management class.
files	Specifies a list of file resources required by the management class.

3.2.7 cobbler package

Package resources are managed using `cobbler package add`

Actions:

Name	Description
install	Install the package. [Default]
uninstall	Uninstall the package.

Attributes:

Name	Description
installer	Which package manager to use, vaild options [rpm|yum].
version	Which version of the package to install.

Example:

```
$ cobbler package add --name=string --comment=string [--action=install|uninstall] -
↪-installer=string [--version=string]
```

3.2.8 cobbler file

Actions:

Name	Description
create	Create the file. [Default]
remove	Remove the file.

Attributes:

Name	Description
mode	Permission mode (as in chmod).
group	The group owner of the file.
user	The user for the file.
path	The path for the file.
template	The template for the file.

Example:

```
$ cobbler file add --name=string --comment=string [--action=string] --mode=string -
↪-group=string --owner=string --path=string [--template=string]
```

3.2.9 cobbler aclsetup

Example:

```
$ cobbler aclsetup
```

3.2.10 cobbler buildiso

Example:

```
$ cobbler buildiso
```

3.2.11 cobbler import

Example:

```
$ cobbler import
```

3.2.12 cobbler list

This list all the names grouped by type. Identically to `cobbler report` there are subcommands for most of the other cobbler commands. (Currently: distro, profile, system, repo, image, mgmtclass, package, file)

```
$ cobbler list
```

3.2.13 cobbler replicate

Cobbler can replicate configurations from a master cobbler server. Each cobbler server is still expected to have a locally relevant `/etc/cobbler/cobbler.conf` and `modules.conf`, as these files are not synced.

This feature is intended for load-balancing, disaster-recovery, backup, or multiple geography support.

Cobbler can replicate data from a central server.

Objects that need to be replicated should be specified with a pattern, such as `--profiles="webservers* dbservers*"` or `--systems="*.example.org"`. All objects matched by the pattern, and all dependencies of those objects matched by the pattern (recursively) will be transferred from the remote server to the central server. This is to say if you intend to transfer `*.example.org` and the definition of the systems have not changed, but a profile above them has changed, the changes to that profile will also be transferred.

In the case where objects are more recent on the local server, those changes will not be overridden locally.

Common data locations will be rsync'ed from the master server unless `--omit-data` is specified.

To delete objects that are no longer present on the master server, use `--prune`.

Warning: This will delete all object types not present on the remote server from the local server, and is recursive. If you use prune, it is best to manage cobbler centrally and not expect changes made on the slave servers to be preserved. It is not currently possible to just prune objects of a specific type.

Example:

```
$ cobbler replicate --master=cobbler.example.org [--distros=pattern] [--
↪profiles=pattern] [--systems=pattern] [--repos-pattern] [--images=pattern] [--
↪prune] [--omit-data]
```

3.2.14 cobbler report

This lists all configuration which cobbler can obtain from the saved data. There are also `report` subcommands for most of the other cobbler commands. (Currently: distro, profile, system, repo, image, mgmtclass, package, file)

```
$ cobbler report --name=[object-name]
```

–name=[object-name]

Optional parameter which filters for object with the given name.

3.2.15 cobbler reposync

Example:

```
$ cobbler reposync
```

3.2.16 cobbler sync

The sync command is very important, though very often unnecessary for most situations. It's primary purpose is to force a rewrite of all configuration files, distribution files in the TFTP root, and to restart managed services. So why is it unnecessary? Because in most common situations (after an object is edited, for example), Cobbler executes what is known as a "lite sync" which rewrites most critical files.

When is a full sync required? When you are using `manage_dhcpd` (Managing DHCP) with systems that use static leases. In that case, a full sync is required to rewrite the `dhcpd.conf` file and to restart the dhcpd service.

Cobbler sync is used to repair or rebuild the contents `/tftpboot` or `/var/www/cobbler` when something has changed behind the scenes. It brings the filesystem up to date with the configuration as understood by cobbler.

Sync should be run whenever files in `/var/lib/cobbler` are manually edited (which is not recommended except for the settings file) or when making changes to automatic installation files. In practice, this should not happen often, though running sync too many times does not cause any adverse effects.

If using cobbler to manage a DHCP and/or DNS server (see the advanced section of this manpage), sync does need to be run after systems are added to regenerate and reload the DHCP/DNS configurations.

The sync process can also be kicked off from the web interface.

Example:

```
$ cobbler sync
```

3.2.17 cobbler validate-autoinstalls

Example:

```
$ cobbler validate-autoinstalls
```

3.2.18 cobbler version

Example:

```
$ cobbler version
```

3.2.19 cobbler signature

Example:

```
$ cobbler signature
```

3.2.20 cobbler get-loaders

Example:

```
$ cobbler get-loaders
```

3.2.21 cobbler hardlink

Example:

```
$ cobbler hardlink
```

3.3 EXIT_STATUS

cobbler's command line returns a zero for success and non-zero for failure.

3.4 Additional Help

We have a Gitter Channel and you also can ask questions as Github-Issues. The IRC Channel on Freenode (#cobbler) is not that active but sometimes there are people who can help you.

The way we would prefer are Github-Issues as they are easily searchable.

CHAPTER 4

Cobblerd

cobbler - a provisioning and update server

4.1 Preamble

We will reefer to cobblerd here as "cobbler" because cobblerd is short for cobbler-daemon which is basically the server. The CLI will be referred to as Cobbler-CLI and Koan as Koan.

4.2 Description

Cobbler manages provisioning using a tiered concept of Distributions, Profiles, Systems, and (optionally) Images and Repositories.

Distributions contain information about what kernel and initrd are used, plus metadata (required kernel parameters, etc).

Profiles associate a Distribution with an automated installation template file and optionally customize the metadata further.

Systems associate a MAC, IP, and other networking details with a profile and optionally customize the metadata further.

Repositories contain yum mirror information. Using cobbler to mirror repositories is an optional feature, though provisioning and package management share a lot in common.

Images are a catch-all concept for things that do not play nicely in the "distribution" category. Most users will not need these records initially and these are described later in the document.

The main advantage of cobbler is that it glues together many disjoint technologies and concepts and abstracts the user from the need to understand them. It allows the systems administrator to concentrate on what he needs to do, and not how it is done.

This manpage will focus on the cobbler command line tool for use in configuring cobbler. There is also mention of the Cobbler WebUI which is usable for day-to-day operation of Cobbler once installed/configured. Docs on the API and XMLRPC components are available online at https://cobbler.github.io or https://cobbler.readthedocs.io

Most users will be interested in the Web UI and should set it up, though the command line is needed for initial configuration – in particular `cobbler check` and `cobbler import`, as well as the repo mirroring features. All of these are described later in the documentation.

4.3 Setup

After installing, run `cobbler check` to verify that cobbler's ecosystem is configured correctly. Cobbler check will direct you on how to modify it's config files using a text editor.

Any problems detected should be corrected, with the potential exception of DHCP related warnings where you will need to use your judgement as to whether they apply to your environment. Run `cobbler sync` after making any changes to the configuration files to ensure those changes are applied to the environment.

It is especially important that the server name field be accurate in `/etc/cobbler/settings`, without this field being correct, automatic installation trees will not be found, and automated installations will fail.

For PXE, if DHCP is to be run from the cobbler server, the dhcp configuration file should be changed as suggested by `cobbler check`. If DHCP is not run locally, the `next-server` field on the DHCP server should at minimum point to the cobbler server's IP and the filename should be set to `pxelinux.0`. Alternatively, cobbler can also generate your dhcp configuration file if you want to run dhcp locally – this is covered in a later section. If you don't already have a DHCP setup managed by some other tool, allowing cobbler to manage your DHCP environment will prove to be useful as it can manage DHCP reservations and other data. If you already have a DHCP setup, moving an existing setup to be managed from within cobbler is relatively painless – though usage of the DHCP management feature is entirely optional. If you are not interested in network booting via PXE and just want to use koan to install virtual systems or replace existing ones, DHCP configuration can be totally ignored. Koan also has a live CD (see koan's manpage) capability that can be used to simulate PXE environments.

4.4 Autoinstallation (Autoyast/Kickstart)

For help in building kickstarts, try using the `system-config-kickstart` tool, or install a new system and look at the `/root/anaconda-ks.cfg` file left over from the installer. General kickstart questions can also be asked at kickstart-list@redhat.com. Cobbler ships some autoinstall templates in /etc/cobbler that may also be helpful.

For autoyast guides and help please reefer to the opensuse project

Also see the website or documentation for additional documentation, user contributed tips, and so on.

4.5 Options

-B –daemonize If you pass no options this is the default one. The Cobbler-Server runs in the background.

-F –no-daemonize The Cobbler-Server runs in the foreground.

-f –log-file Choose a destination for the logfile (currently has no effect).

-l –log-level Choose a loglevel for the application (currently has no effect).

CHAPTER 5

Cobbler Configuration

There are two main settings files: settings and modules.conf. Both files can be found under `/etc/cobbler/` and both are written in YAML.

5.1 settings

5.1.1 allow_duplicate_hostnames

if 1, cobbler will allow insertions of system records that duplicate the `--dns-name` information of other system records. In general, this is undesirable and should be left 0.

default: `0`

5.1.2 allow_duplicate_ips

if 1, cobbler will allow insertions of system records that duplicate the ip address information of other system records. In general, this is undesirable and should be left 0.

default: `0`

5.1.3 allow_duplicate_macs

If 1, cobbler will allow insertions of system records that duplicate the mac address information of other system records. In general, this is undesirable.

default: `0`

5.1.4 allow_dynamic_settings

If 1, cobbler will allow settings to be changed dynamically without a restart of the cobblerd daemon. You can only change this variable by manually editing the settings file, and you MUST restart cobblerd after changing it.

default: `0`

5.1.5 anamon_enabled

By default, installs are *not* set to send installation logs to the cobbler server. With `anamon_enabled`, automatic installation templates may use the `pre_anamon` snippet to allow remote live monitoring of their installations from the cobbler server. Installation logs will be stored under `/var/log/cobbler/anamon/`.

Note: This does allow an xmlrpc call to send logs to this directory, without authentication, so enable only if you are ok with this limitation.

default: `0`

5.1.6 authn_pam_service

If using authn_pam in the `modules.conf`, this can be configured to change the PAM service authentication will be tested against.

default: `"login"`

5.1.7 auth_token_expiration

How long the authentication token is valid for, in seconds.

default: `3600`

5.1.8 autoinstall_snippets_dir

This is a directory of files that cobbler uses to make templating easier. See the Wiki for more information. Changing this directory should not be required.

default: `/var/lib/cobbler/snippets`

5.1.9 autoinstall_templates_dir

This is a directory of files that cobbler uses to make templating easier. See the Wiki for more information. Changing this directory should not be required.

default: `/var/lib/cobbler/templates`

5.1.10 boot_loader_conf_template_dir

Location of templates used for boot loader config generation.

default: `"/etc/cobbler/boot_loader_conf"`

5.1.11 build_reporting_*

Email out a report when cobbler finishes installing a system.

- enabled: set to 1 to turn this feature on
- sender: optional
- email: which addresses to email
- smtp_server: used to specify another server for an MTA
- subject: use the default subject unless overridden

defaults:

```
build_reporting_enabled: 0
build_reporting_sender: ""
build_reporting_email: [ 'root@localhost' ]
build_reporting_smtp_server: "localhost"
build_reporting_subject: ""
build_reporting_ignorelist: [ "" ]
```

5.1.12 cheetah_import_whitelist

Cheetah-language autoinstall templates can import Python modules. while this is a useful feature, it is not safe to allow them to import anything they want. This whitelists which modules can be imported through Cheetah. Users can expand this as needed but should never allow modules such as subprocess or those that allow access to the filesystem as Cheetah templates are evaluated by cobblerd as code.

default:

- "random"
- "re"
- "time"
- "netaddr"

5.1.13 createrepo_flags

Default createrepo_flags to use for new repositories. If you have `createrepo >= 0.4.10`, consider `-c cache --update -C`, which can dramatically improve your `cobbler reposync` time. `-s sha` enables working with Fedora repos from F11/F12 from EL-4 or EL-5 without python-hashlib installed (which is not available on EL-4)

default: `"-c cache -s sha"`

5.1.14 default_autoinstall

If no autoinstall template is specified to profile add, use this template.

default: `/var/lib/cobbler/autoinstall_templates/default.ks`

5.1.15 default_name_*

Configure all installed systems to use these nameservers by default unless defined differently in the profile. For DHCP configurations you probably do /not/ want to supply this.

defaults:

```
default_name_servers: []
default_name_servers_search: []
```

5.1.16 default_ownership

if using the `authz_ownership` module (see the Wiki), objects created without specifying an owner are assigned to this owner and/or group. Can be a comma seperated list.

default:

- "admin"

5.1.17 default_password_crypted

Cobbler has various sample automatic installation templates stored in `/var/lib/cobbler/autoinstall_templates/`. This controls what install (root) password is set up for those systems that reference this variable. The factory default is "cobbler" and cobbler check will warn if this is not changed. The simplest way to change the password is to run `openssl passwd -1` and put the output between the `""`.

default: `"$1$mF86/UHC$WvcIcX2t6crBz2onWxyac."`

5.1.18 default_template_type

The default template type to use in the absence of any other detected template. If you do not specify the template with `#template=<template_type>` on the first line of your templates/snippets, cobbler will assume try to use the following template engine to parse the templates.

Current valid values are: cheetah, jinja2

default: `"cheetah"`

5.1.19 default_virt_bridge

For libvirt based installs in koan, if no virt-bridge is specified, which bridge do we try? For EL 4/5 hosts this should be `xenbr0`, for all versions of Fedora, try `virbr0`. This can be overriden on a per-profile basis or at the koan command line though this saves typing to just set it here to the most common option.

default: `xenbr0`

5.1.20 default_virt_file_size

Use this as the default disk size for virt guests (GB).

default: `5`

5.1.21 default_virt_ram

Use this as the default memory size for virt guests (MB).

default: `512`

5.1.22 default_virt_type

If koan is invoked without `--virt-type` and no virt-type is set on the profile/system, what virtualization type should be assumed?

Current valid values are: xenpv, xenfv, qemu, vmware

NOTE: this does not change what `virt_type` is chosen by import.

default: `xenpv`

5.1.23 enable_gpxe

Enable gPXE booting? Enabling this option will cause cobbler to copy the `undionly.kpxe` file to the tftp root directory, and if a profile/system is configured to boot via gpxe it will chain load off `pxelinux.0`.

default: `0`

5.1.24 enable_menu

Controls whether cobbler will add each new profile entry to the default PXE boot menu. This can be over-ridden on a per-profile basis when adding/editing profiles with `--enable-menu=0/1`. Users should ordinarily leave this setting enabled unless they are concerned with accidental reinstalls from users who select an entry at the PXE boot menu. Adding a password to the boot menus templates may also be a good solution to prevent unwanted reinstallations

default: `1`

5.1.25 http_port

Change this port if Apache is not running plaintext on port 80. Most people can leave this alone.

default: `80`

5.1.26 kernel_options

Kernel options that should be present in every cobbler installation. Kernel options can also be applied at the distro/profile/system level.

default: `{}`

5.1.27 ldap_*

Configuration options if using the authn_ldap module. See the the Wiki for details. This can be ignored if you are not using LDAP for WebUI/XMLRPC authentication.

defaults:

```
ldap_server: "ldap.example.com"
ldap_base_dn: "DC=example,DC=com"
ldap_port: 389
ldap_tls: 1
ldap_anonymous_bind: 1
ldap_search_bind_dn: ''
ldap_search_passwd: ''
ldap_search_prefix: 'uid='
ldap_tls_cacertfile: ''
ldap_tls_keyfile: ''
ldap_tls_certfile: ''
```

5.1.28 mgmt_*

Cobbler has a feature that allows for integration with config management systems such as Puppet. The following parameters work in conjunction with `--mgmt-classes` and are described in further detail at: https://github.com/cobbler/cobbler/wiki/Using-cobbler-with-a-configuration-management-system

```
mgmt_classes: []
mgmt_parameters:
    from_cobbler: 1
```

5.1.29 puppet_auto_setup

If enabled, this setting ensures that puppet is installed during machine provision, a client certificate is generated and a certificate signing request is made with the puppet master server.

default: `0`

5.1.30 sign_puppet_certs_automatically

When puppet starts on a system after installation it needs to have its certificate signed by the puppet master server. Enabling the following feature will ensure that the puppet server signs the certificate after installation if the puppet master server is running on the same machine as cobbler. This requires `puppet_auto_setup` above to be enabled.

default: `0`

5.1.31 puppetca_path

Location of the puppet executable, used for revoking certificates.

default: `"/usr/bin/puppet"`

5.1.32 remove_old_puppet_certs_automatically

When a puppet managed machine is reinstalled it is necessary to remove the puppet certificate from the puppet master server before a new certificate is signed (see above). Enabling the following feature will ensure that the certificate for the machine to be installed is removed from the puppet master server if the puppet master server is running on the same machine as cobbler. This requires `puppet_auto_setup` above to be enabled

default: `0`

5.1.33 puppet_server

Choose a `--server` argument when running puppetd/puppet agent during autoinstall. This one is commented out by default.

default: `'puppet'`

5.1.34 puppet_version

Let cobbler know that you're using a newer version of puppet. Choose version 3 to use: 'puppet agent'; version 2 uses status quo: 'puppetd'. This one is commented out by default.

default: `2`

5.1.35 puppet_parameterized_classes

Choose whether to enable puppet parameterized classes or not. Puppet versions prior to 2.6.5 do not support parameters. This one is commented out by default.

default: 1

5.1.36 manage_dhcp

Set to 1 to enable Cobbler's DHCP management features. The choice of DHCP management engine is in `/etc/cobbler/modules.conf`

default: `0`

5.1.37 manage_dns

Set to 1 to enable Cobbler's DNS management features. The choice of DNS mangement engine is in `/etc/cobbler/modules.conf`

default: `0`

5.1.38 bind_chroot_path

Set to path of bind chroot to create bind-chroot compatible bind configuration files. This should be automatically detected.

default: `""`

5.1.39 bind_master

Set to the ip address of the master bind DNS server for creating secondary bind configuration files.

default: `127.0.0.1`

5.1.40 manage_tftpd

Set to 1 to enable Cobbler's TFTP management features. the choice of TFTP mangement engine is in `/etc/cobbler/modules.conf`

default: `1`

5.1.41 tftpboot_location

This variable contains the location of the tftpboot directory. If this directory is not present cobbler does not start.

Default: `/srv/tftpboot`

5.1.42 manage_rsync

Set to 1 to enable Cobbler's RSYNC management features.

default: `0`

5.1.43 manage_*

If using BIND (named) for DNS management in `/etc/cobbler/modules.conf` and manage_dns is enabled (above), this lists which zones are managed. See the Wiki (https://github.com/cobbler/cobbler/wiki/Dns-management) for more info

defaults:

```
manage_forward_zones: []
manage_reverse_zones: []
```

5.1.44 next_server

If using cobbler with `manage_dhcp`, put the IP address of the cobbler server here so that PXE booting guests can find it. If you do not set this correctly, this will be manifested in TFTP open timeouts.

default: `127.0.0.1`

5.1.45 power_management_default_type

Settings for power management features. These settings are optional. See https://github.com/cobbler/cobbler/wiki/Power-management to learn more.

Choices (refer to codes.py):

- apc_snmp
- bladecenter
- bullpap
- drac
- ether_wake
- ilo
- integrity
- ipmilan
- ipmitool
- lpar
- rsa
- virsh
- wti

default: `ipmitool`

5.1.46 pxe_just_once

If this setting is set to 1, cobbler systems that pxe boot will request at the end of their installation to toggle the `--netboot-enabled` record in the cobbler system record. This eliminates the potential for a PXE boot loop if the system is set to PXE first in it's BIOS order. Enable this if PXE is first in your BIOS boot order, otherwise leave this disabled. See the manpage for `--netboot-enabled`.

default: `1`

5.1.47 nopxe_with_triggers

If this setting is set to one, triggers will be executed when systems will request to toggle the `--netboot-enabled` record at the end of their installation.

default: `1`

5.1.48 redhat_management_server

This setting is only used by the code that supports using Spacewalk/Satellite authentication within Cobbler Web and Cobbler XMLRPC.

default: `"xmlrpc.rhn.redhat.com"`

5.1.49 redhat_management_permissive

If using `authn_spacewalk` in `modules.conf` to let cobbler authenticate against Satellite/Spacewalk's auth system, by default it will not allow per user access into Cobbler Web and Cobbler XMLRPC. In order to permit this, the following setting must be enabled HOWEVER doing so will permit all Spacewalk/Satellite users of certain types to edit all of cobbler's configuration. these roles are: `config_admin` and `org_admin`. Users should turn this on only if they want this behavior and do not have a cross-multi-org seperation concern. If you have a single org in your satellite, it's probably safe to turn this on and then you can use CobblerWeb alongside a Satellite install.

default: `0`

5.1.50 redhat_management_key

Specify the default Red Hat authorization key to use to register system. If left blank, no registration will be attempted. Similarly you can set the `--redhat-management-key` to blank on any system to keep it from trying to register.

default: `""`

5.1.51 register_new_installs

If set to `1`, allows `/usr/bin/cobbler-register` (part of the koan package) to be used to remotely add new cobbler system records to cobbler. This effectively allows for registration of new hardware from system records.

default: `0`

5.1.52 reposync_flags

Flags to use for yum's reposync. If your version of yum reposync does not support `-l`, you may need to remove that option.

default: `"-l -n -d"`

5.1.53 restart_*

When DHCP and DNS management are enabled, `cobbler sync` can automatically restart those services to apply changes. The exception for this is if using ISC for DHCP, then omapi eliminates the need for a restart. `omapi`, however, is experimental and not recommended for most configurations. If DHCP and DNS are going to be managed, but hosted on a box that is not on this server, disable restarts here and write some other script to ensure that the config files get copied/rsynced to the destination box. This can be done by modifying the restart services trigger. Note that if `manage_dhcp` and `manage_dns` are disabled, the respective parameter will have no effect. Most users should not need to change this.

defaults:

```
restart_dns: 1
restart_dhcp: 1
```

5.1.54 run_install_triggers

Install triggers are scripts in `/var/lib/cobbler/triggers/install` that are triggered in autoinstall pre and post sections. Any executable script in those directories is run. They can be used to send email or perform other actions. They are currently run as root so if you do not need this functionality you can disable it, though this will also disable `cobbler status` which uses a logging trigger to audit install progress.

default: `1`

5.1.55 scm_track_*

enables a trigger which version controls all changes to `/var/lib/cobbler` when add, edit, or sync events are performed. This can be used to revert to previous database versions, generate RSS feeds, or for other auditing or backup purposes. Git and Mercurial are currently supported, but Git is the recommend SCM for use with this feature.

default:

```
scm_track_enabled: 0
scm_track_mode: "git"
scm_track_author: "cobbler <cobbler@localhost>"
scm_push_script: "/bin/true"
```

5.1.56 server

This is the address of the cobbler server – as it is used by systems during the install process, it must be the address or hostname of the system as those systems can see the server. if you have a server that appears differently to different subnets (dual homed, etc), you need to read the `--server-override` section of the manpage for how that works.

default: `127.0.0.1`

5.1.57 client_use_localhost

If set to 1, all commands will be forced to use the localhost address instead of using the above value which can force commands like cobbler sync to open a connection to a remote address if one is in the configuration and would traceback.

default: `0`

5.1.58 client_use_https

If set to 1, all commands to the API (not directly to the XMLRPC server) will go over HTTPS instead of plaintext. Be sure to change the `http_port` setting to the correct value for the web server.

default: `0`

5.1.59 virt_auto_boot

Should new profiles for virtual machines default to auto booting with the physical host when the physical host reboots? This can be overridden on each profile or system object.

default: `1`

5.1.60 webdir

Cobbler's web directory. Don't change this setting – see the Wiki on "relocating your cobbler install" if your /var partition is not large enough.

default: `@@webroot@@/cobbler`

5.1.61 webdir_whitelist

Directories that will not get wiped and recreated on a `cobbler sync`.

default:

```
webdir_whitelist:
  - misc
  - web
  - webui
  - localmirror
  - repo_mirror
  - distro_mirror
  - images
  - links
  - pub
  - repo_profile
  - repo_system
  - svc
  - rendered
  - .link_cache
```

5.1.62 xmlrpc_port

Cobbler's public XMLRPC listens on this port. Change this only if absolutely needed, as you'll have to start supplying a new port option to koan if it is not the default.

default: `25151`

5.1.63 yum_post_install_mirror

`cobbler repo add` commands set cobbler up with repository information that can be used during autoinstall and is automatically set up in the cobbler autoinstall templates. By default, these are only available at install time. To make these repositories usable on installed systems (since cobbler makes a very convenient mirror) set this to 1. Most users can safely set this to 1. Users who have a dual homed cobbler server, or are installing laptops that will not always have access to the cobbler server may wish to leave this as 0. In that case, the cobbler mirrored yum repos are still accessible at `http://cobbler.example.org/cblr/repo_mirror` and yum configuration can still be done manually. This is just a shortcut.

default: `1`

5.1.64 yum_distro_priority

The default yum priority for all the distros. This is only used if yum-priorities plugin is used. 1 is the maximum value. Tweak with caution.

default: `1`

5.1.65 yumdownloader_flags

Flags to use for yumdownloader. Not all versions may support `--resolve`.

default: `"--resolve"`

5.1.66 serializer_pretty_json

Sort and indent JSON output to make it more human-readable.

default: `0`

5.1.67 replicate_rsync_options

replication rsync options for distros, autoinstalls, snippets set to override default value of `-avzH`

default: `"-avzH"`

5.1.68 replicate_repo_rsync_options

Replication rsync options for repos set to override default value of `-avzH`

default: `"-avzH"`

5.1.69 always_write_dhcp_entries

Always write DHCP entries, regardless if netboot is enabled.

default: `0`

5.1.70 proxy_url_ext:

External proxy - used by: get-loaders, reposync, signature update. Per default commented out.

defaults:

```
http: http://192.168.1.1:8080
https: https://192.168.1.1:8443
```

5.1.71 proxy_url_int

Internal proxy - used by systems to reach cobbler for kickstarts.

E.g.: proxy_url_int: `http://10.0.0.1:8080`

default: `""`

5.1.72 jinja2_includedir

This is a directory of files that cobbler uses to include files into Jinja2 templates. Per default this settings is commented out.

default: `/var/lib/cobbler/jinja2`

5.1.73 include

Include other configuration snippets with this regular expresion.

default: `[ "/etc/cobbler/settings.d/*.settings" ]`

5.2 modules.conf

If you have own custom modules which are not shipped with Cobbler directly you may have additional sections here.

5.2.1 authentication

What users can log into the WebUI and Read-Write XMLRPC?

Choices:

- authn_denyall – no one (default)
- authn_configfile – use /etc/cobbler/users.digest (for basic setups)
- authn_passthru – ask Apache to handle it (used for kerberos)
- authn_ldap – authenticate against LDAP
- authn_spacewalk – ask Spacewalk/Satellite (experimental)
- authn_pam – use PAM facilities
- authn_testing – username/password is always testing/testing (debug)
- (user supplied) – you may write your own module

WARNING: this is a security setting, do not choose an option blindly.

For more information:

- https://github.com/cobbler/cobbler/wiki/Cobbler-web-interface
- https://github.com/cobbler/cobbler/wiki/Security-overview
- https://github.com/cobbler/cobbler/wiki/Kerberos
- https://github.com/cobbler/cobbler/wiki/Ldap

default: `authn_configfile`

5.2.2 authorization

Once a user has been cleared by the WebUI/XMLRPC, what can they do?

Choices:

- authz_allowall – full access for all authneticated users (default)
- authz_ownership – use users.conf, but add object ownership semantics
- (user supplied) – you may write your own module

WARNING: this is a security setting, do not choose an option blindly. If you want to further restrict cobbler with ACLs for various groups, pick authz_ownership. authz_allowall does not support ACLs. configfile does but does not support object ownership which is useful as an additional layer of control.

For more information:

- https://github.com/cobbler/cobbler/wiki/Cobbler-web-interface
- https://github.com/cobbler/cobbler/wiki/Security-overview
- https://github.com/cobbler/cobbler/wiki/Web-authorization

default: `authz_allowall`

5.2.3 dns

Chooses the DNS management engine if manage_dns is enabled in `/etc/cobbler/settings`, which is off by default.

Choices:

- manage_bind – default, uses BIND/named
- manage_dnsmasq – uses dnsmasq, also must select dnsmasq for dhcp below
- manage_ndjbdns – uses ndjbdns

NOTE: More configuration is still required in `/etc/cobbler`

For more information: https://github.com/cobbler/cobbler/wiki/Dns-management

default: `manage_bind`

5.2.4 dhcp

Chooses the DHCP management engine if `manage_dhcp` is enabled in `/etc/cobbler/settings`, which is off by default.

Choices:

- manage_isc – default, uses ISC dhcpd
- manage_dnsmasq – uses dnsmasq, also must select dnsmasq for dns above

NOTE: More configuration is still required in `/etc/cobbler`

For more information: https://github.com/cobbler/cobbler/wiki/Dhcp-management

default: `manage_isc`

5.2.5 tftpd

Chooses the TFTP management engine if manage_tftp is enabled in `/etc/cobbler/settings`, which is ON by default.

Choices:

- manage_in_tftpd – default, uses the system's tftp server
- manage_tftpd_py – uses cobbler's tftp server

default: `manage_in_tftpd`

CHAPTER 6

User Guide

6.1 Web-Interface

Please be patient until we have time to rework this section or please file a PR for this section.

The standard login for the Web-UI can be read below. We would recommend to change this as soon as possible!

Username: `cobbler` Password: `cobbler`

6.1.1 Old Release 2.8.x

https://cobbler.readthedocs.io/en/release28/web-interface.html

6.1.2 Old Github-Wiki Entry

Most of the day-to-day actions in cobbler's command line can be performed in Cobbler's Web UI.

With the web user interface (WebUI), you can:

- View all of the cobbler objects and the settings
- Add and delete a system, distro, profile, or system
- Run the equivalent of a `cobbler sync`
- Edit kickstart files (which must be in `/etc/cobbler` and `/var/lib/cobbler/kickstarts`)

You cannnot (yet):

- Auto-Import media
- Auto-Import a rsync mirror of install trees
- Do a `cobbler reposync` to mirror or update yum content
- Do a `cobbler validateks`

The WebUI can be very good for day-to-day configuring activities, but the CLI is still required for basic bootstrapping and certain other activities.

The WebUI is intended to be self-explanatory and contains tips and explanations for nearly every field you can edit. It also contains links to additional documentation, including the Cobbler manpage documentation in HTML format.

Who logs in and what they can access is controlled by Web Authentication and Web Authorization. The default options are mostly good for getting started, but for safety reasons the default authentication is "denyall" so you will at least need to address that.

Basic Setup

1. You must have installed the cobbler-web package
2. Your `/etc/httpd/conf.d/cobbler_web.conf` should look something like this:

```
# This configuration file enables the cobbler web interface (django version)
# Force everything to go to https
RewriteEngine on
RewriteCond %{HTTPS} off
RewriteCond %{REQUEST_URI} ^/cobbler_web
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}

WSGIScriptAlias /cobbler_web /usr/share/cobbler/web/cobbler.wsgi

# The following Directory Entry in Apache Configs solves 403 Forbidden errors.
<Directory "/usr/share/cobbler/web">
  Order allow,deny
  Allow from all
</Directory>

# Display Cobbler Themes + Logo graphics.
<Directory "/var/www/cobbler_webui_content">
Order allow,deny
Allow from all
</Directory>
```

3. Your `/etc/cobbler/modules.conf` should look something like this:

```
[authentication]
module = authn_configfile

[authorization]
module = authz_allowall
```

4. Change the password for the 'cobbler' username:

```
htdigest /etc/cobbler/users.digest "Cobbler" cobbler
```

5. If this is not a new install, your Apache configuration for Cobbler might not be current.

```
cp /etc/httpd/conf.d/cobbler.conf.rpmnew /etc/httpd/conf.d/cobbler.conf
```

6. Now restart Apache and Cobblerd

```
/sbin/service cobblerd restart
/sbin/service httpd restart
```

7. If you use SELinux, you may also need to set the following, so that the WebUI can connect with the XMLRPC:

```
setsebool -P httpd_can_network_connect true
```

Basic setup (2.2.x and higher)

In addition to the steps above, cobbler 2.2.x has a requirement for `mod_wsgi` which, when installed via EPEL, will be disabled by default. Attempting to start httpd will result in:

```
Invalid command 'WSGIScriptAliasMatch', perhaps misspelled \
  or defined by a module not included in the server configuration
```

You can enable this module by editing `/etc/httpd/conf.d/wsgi.conf` and un-commenting the "LoadModule wsgi_module modules/mod_wsgi.so" line.

Next steps

It should be ready to go. From your web browser visit the URL on your bootserver that resembles:

```
https://bootserver.example.com/cobbler_web
```

and log in with the username (usually cobbler) and password that you set earlier.

Should you ever need to debug things, see the following log files:

```
/var/log/httpd/error_log
/var/log/cobbler/cobbler.log
```

Further setup

Cobbler authenticates all WebUI logins through `cobblerd`, which uses a configurable authentication mechanism. You may wish to adjust that for your environment. For instance, if in `modules.conf` above you choose to stay with the `authentication.configfile` module, you may want to add your system administrator usernames to the digest file. To do this it is recommended to use either `openssl` or Python directly.

Example using `openssl 1.1.1` or later:

```
printf "foobar" | openssl dgst -sha3-512
```

It is possible with `openssl` to generate hashes for the following hash algorithms which are configurable: blake2b512, blake2s256, shake128, shake256, sha3-224m sha3-256, sha3-384, sha3-512

Example using Python (using the python interactive shell):

```
import hashlib
hashlib.sha3_512("<PASSWORD>".encode('utf-8')).hexdigest()
```

Python of course will always have all possible hash algorithms available which are valid in the context of Cobbler.

Both examples return the same result when executed with the same password. The file itself is structured according to the following: `<USERNAME>:<REALM>:<PASSWORDHASH>`. Normally `<REALM>` will be `Cobbler`. Other values are currently not valid. Please add the user, realm and passwordhash with your preferred editor. Normally there should be no need to restart cobbler when a new user is added, removed or the password is changed. The authentication process reads the file every time a user is authenticated.

You may also want to refine for authorization settings.

Before Cobbler 3.1.2 it was recommended to do edit the file `users.digest` with the following command. Since `md5` is not FIPS compatible from Cobbler 3.1.3 and onwards this is not possible anymore. The file was also just read once per Cobbler start and thus a change of the data requires that Cobbler is restarted that it picks up these changes.

```
htdigest /etc/cobbler/users.digest "Cobbler" <username>
```

Rewrite Rule for secure-http

To redirect access to the WebUI via https on an Apache webserver, you can use the following rewrite rule, probably at the end of Apache's `ssl.conf`:

```
### Force SSL only on the WebUI
<VirtualHost *:80>
    <LocationMatch "^/cobbler_web/*">
       RewriteEngine on
       RewriteRule ^(.*) https://%{SERVER_NAME}/%{REQUEST_URI} [R,L]
   </LocationMatch>
</VirtualHost>
```

6.2 Configuration Management Integrations

Cobbler contains features for integrating an installation environment with a configuration management system, which handles the configuration of the system after it is installed by allowing changes to configuration files and settings.

Resources are the lego blocks of configuration management. Resources are grouped together via Management Classes, which are then linked to a system. Cobbler supports two (2) resource types. Resources are configured in the order listed below.

The initial provisioning of client systems with cobbler is just one component of their management. We also need to consider how to continue to manage them using a configuration management system (CMS). Cobbler can help you provision and introduce a CMS onto your client systems.

One option is cobbler's own lightweight CMS. For that, see the document *Built-In Configuration Management*.

Here we discuss the other option: deploying a CMS such as cfengine3, puppet, bcfg2, Chef, etc.

Cobbler doesn't force you to chose a particular CMS (or to use one at all), though it helps if you do some things to link cobbler's profiles with the "profiles" of the CMS. This, in general, makes management of both a lot easier.

Note that there are two independent "variables" here: the possible client operating systems and the possible CMSes. We don't attempt to cover all details of all combinations; rather we illustrate the principles and give a small number of illustrative examples of particular OS/CMS combinations. Currently cobbler has better support for Redhat-based OSes and for Puppet so the current examples tend to deal with this combination.

6.2.1 Background considerations

Machine lifecycle

A typical computer has a lifecycle something like:

- installation
- initial configuration
- ongoing configuration and maintenance
- decommissioning

Typically installation happens once. Likewise, the initial configuration happens once, usually shortly after installation. By contrast ongoing configuration evolves over an extended period, perhaps of several years. Sometimes part of that ongoing configuration may involve re-installing an OS from scratch. We can regard this as repeating the earlier phase.

We need not consider decommissioning here.

Installation clearly belongs (in our context) to Cobbler. In a complementary manner, ongoing configuration clearly belongs to the CMS. But what about initial configuration?

Some sites consider their initial configuration as the final phase of installation: in our context, that would put it at the back end of Cobbler, and potentially add significant configuration-based complication to the installation-based Cobbler set-up.

But it is worth considering initial configuration as the first step of ongoing configuration: in our context that would put it as part of the CMS, and keep the Cobbler set-up simple and uncluttered.

Local package repositories

Give consideration to:

- local mirrors of OS repositories
- local repository of local packages
- local repository of pick-and-choose external packages

In particular consider having the packages for your chosen CMS in one of the latter.

Package management

Some sites set up Cobbler always to deploy just a minimal subset of packages, then use the CMS to install many others in a large-scale fashion. Other sites may set up Cobbler to deploy tailored sets of packages to different types of machines, then use the CMS to do relatively small-scale fine-tuning of that.

6.2.2 General scheme

We need to consider getting Cobbler to install and automatically invoke the CMS software.

Set up Cobbler to include a package repository that contains your chosen CMS:

```
cobbler repo add ...
```

Then (illustrating a Redhat/Puppet combination) set up the kickstart file to say something like:

```
%packages
puppet

%post
/sbin/chkconfig --add puppet
```

The detail may need to be more substantial, requiring some other associated local packages, files and configuration. You may wish to manage this through [Kickstart snippets](Kickstart Snippets).

David Lutterkort has a walkthrough for kickstart. While his example is written for Redhat (Fedora) and Puppet, the principles are useful for other OS/CMS combinations.

6.2.3 Built-In Configuration Management

Cobbler is not just an installation server, it can also enable two different types of ongoing configuration management system (CMS):

- integration with an established external CMS such as cfengine3, bcfg2, Chef, or puppet, discussed [elsewhere](Using cobbler with a configuration management system);
- its own, much simpler, lighter-weight, internal CMS, discussed here.

Setting up

Cobbler's internal CMS is focused around packages and templated configuration files, and installing these on client systems.

This all works using the same Cheetah-powered templating engine used in Kickstart Templating, so once you learn about the power of treating your distribution answer files as templates, you can use the same templating to drive your CMS configuration files.

For example:

```
cobbler profile edit --name=webserver --template-files=/srv/cobbler/x.template=/
↪etc/foo.conf
```

A client system installed via the above profile will gain a file `/etc/foo.conf` which is the result of rendering the template given by `/srv/cobbler/x.template`. Multiple files may be specified; each `template=destination` pair should be placed in a space-separated list enclosed in quotes:

```
--template-files="srv/cobbler/x.template=/etc/xfile.conf srv/cobbler/y.template=/
↪etc/yfile.conf"
```

Template files

Because the template files will be parsed by the Cheetah parser, they must conform to the guidelines described in Kickstart Templating. This is particularly important when the file is generated outside a Cheetah environment. Look for, and act on, Cheetah 'ParseError' errors in the Cobbler logs.

Template files follows general Cheetah syntax, so can include Cheetah variables. Any variables you define anywhere in the cobbler object hierarchy (distros, profiles, and systems) are available to your templates. To see all the variables available, use the command:

```
cobbler profile dumpvars --name=webserver
```

Cobbler snippets and other advanced features can also be employed.

Ongoing maintenance

Koan can pull down files to keep a system updated with the latest templates and variables:

```
koan --server=cobbler.example.org --profile=foo --update-files
```

You could also use `--server=bar` to retrieve a more specific set of templating. Koan can also autodetect the server if the MAC address is registered.

Further uses

This Cobbler/Cheetah templating system can serve up templates via the magic URLs (see "Leveraging Mod Python" below). To do this ensure that the destination path given to any `--template-files` element is relative, not absolute; then Cobbler and koan won't download those files.

For example, in:

```
cobbler profile edit --name=foo --template-files="/srv/templates/a.src=/etc/foo/a.
↪conf /srv/templates/b.src=1"
```

Cobbler and koan would automatically download the rendered `a.src` to replace the file `/etc/foo/a.conf`, but the `b.src` file would not be downloaded to anything because the destination pathname `1` is not absolute.

This technique enables using the Cobbler/Cheetah templating system to build things that other systems can fetch and use, for instance, BIOS config files for usage from a live environment.

Leveraging Mod Python

All template files are generated dynamically at run-time. If a change is made to a template, a `--ks-meta` variable or some other variable in cobbler, the result of template rendering will be different on subsequent runs. This is covered in more depth in the Developer documentation.

Possible future developments

- Serving and running scripts via `--update-files` (probably staging them through `/var/spool/koan`).
- Auto-detection of the server name if `--ip` is registered.

6.2.4 Terraform Provider

This is developed and maintained by the Terraform community. You will find more information in the docs under https://www.terraform.io/docs/providers/cobbler/index.html

The code for the Terraform-Provider can be found at: https://github.com/terraform-providers/terraform-provider-cobbler

6.2.5 Ansible

Although we currently can not provide something official we can indeed link some community work here:

- https://github.com/ac427/my_cm
- https://github.com/AnKosteck/ansible-cluster
- https://github.com/osism/ansible-cobbler
- https://github.com/hakoerber/ansible-roles

6.2.6 Saltstack

Although we currently can not provide something official we can indeed link some community work here:

- https://github.com/hakoerber/salt-states/tree/master/cobbler

6.2.7 Vagrant

Although we currently can not provide something official we can indeed link some community work here:

- https://github.com/davegermiquet/vmwarevagrantcobblercentos
- https://github.com/dratushnyy/tools
- https://github.com/mkusanagi/cobbler-kickstart-playground

6.2.8 Puppet

There is also an example of Puppet deploying Cobbler: https://github.com/gothicfann/puppet-cobbler

This example is relatively advanced, involving Cobbler "mgmt-classes" to control different types of initial configuration. But if instead you opt to put most of the initial configuration into the Puppet CMS rather than here, then things could be simpler.

Keeping Class Mappings In Cobbler

First, we assign management classes to distro, profile, or system objects.

```
cobbler distro edit --name=distro1 --mgmt-classes="distro1"
cobbler profile add --name=webserver --distro=distro1 --mgmt-classes="webserver␣
↪likes_llamas" --kickstart=/etc/cobbler/my.ks
cobbler system edit --name=system --profile=webserver --mgmt-classes="orange" --
↪dns-name=system.example.org
```

For Puppet, the `--dns-name` (shown above) must be set because this is what puppet will be sending to cobbler and is how we find the system. Puppet doesn't know about the name of the system object in cobbler. To play it safe you probably want to use the FQDN here (which is also what you want if you were using Cobbler to manage your DNS, which you don't have to be doing).

External Nodes

For more documentation on Puppet's external nodes feature, see https://docs.puppetlabs.com

Cobbler provides one, so configure puppet to use `/usr/bin/cobbler-ext-nodes`:

```
[main]
external_nodes = /usr/bin/cobbler-ext-nodes
```

Note: if you are using puppet 0.24 or later then you will want to also add the following to your configuration file.

```
node_terminus = exec
```

You may wonder what this does. This is just a very simple script that grabs the data at the following URL, which is a URL that always returns a YAML document in the way that Puppet expects it to be returned. This file contains all the parameters and classes that are to be assigned to the node in question. The magic URL being visited is powered by Cobbler.

```
http://cobbler/cblr/svc/op/puppet/hostname/foo
```

(for developer information about this magic URL, visit https://fedorahosted.org/cobbler/wiki/ModPythonDetails)

And this will return data such as:

```
---
classes:
    - distro1
    - webserver
    - likes_llamas
    - orange
parameters:
    tree: 'http://.../x86_64/tree'
```

Where do the parameters come from? Everything that cobbler tracks in `--ks-meta` is also a parameter. This way you can easily add parameters as easily as you can add classes, and keep things all organized in one place.

What if you have global parameters or classes to add? No problem. You can also add more classes by editing the following fields in `/etc/cobbler/settings`:

```
# cobbler has a feature that allows for integration with config management
# systems such as Puppet.  The following parameters work in conjunction with

# --mgmt-classes  and are described in furhter detail at:
# https://fedorahosted.org/cobbler/wiki/UsingCobblerWithConfigManagementSystem
mgmt_classes: []
mgmt_parameters:
   from_cobbler: 1
```

Alternate External Nodes Script

Attached at `puppet_node.py` is an alternate external node script that fills in the nodes with items from a manifests repository (at `/etc/puppet/manifests/`) and networking information from cobbler. It is configured like the above from the puppet side, and then looks for `/etc/puppet/external_node.yaml` for cobbler side configuration. The configuration is as follows.

```
base: /etc/puppet/manifests/nodes
cobbler: <%= cobbler_host %>
no_yaml: puppet::noyaml
no_cobbler: network::nocobbler
bad_yaml: puppet::badyaml
unmanaged: network::unmanaged
```

The output for network information will be in the form of a pseudo data structure that allows puppet to split it apart and create the network interfaces on the node being managed.

6.2.9 cfengine support

Documentation to be added

6.2.10 bcfg2 support

Documentation to be added

6.2.11 Chef

Documentation to be added.

There is some integration information on bootstrapping chef clients with cobbler in [this blog article](http://blog.milford.io/2012/03/getting-a-basic-cobbler-server-going-on-centos/)

6.2.12 Conclusion

Hopefully this should get you started in linking up your provisioning configuration with your CMS implementation. The examples provided are for Puppet, but we can (in the future) presumably extend `--mgmt-classes` to work with other tools... Just let us know what you are interested in, or perhaps take a shot at creating a patch for it.

6.2.13 Attachments

- [puppet_node.py](/cobbler/attachment/wiki/UsingCobblerWithConfigManagementSystem/puppet_node.py) (2.5 kB) -Alternate External Nodes Script, added by shenson on 12/09/10 20:33:36.

6.3 API

Cobbler also makes itself available as an XMLRPC API for use by higher level management software. Learn more at https://cobbler.github.io

6.4 Triggers

Triggers provide a way to integrate cobbler with arbitrary 3rd party software without modifying cobbler's code. When adding a distro, profile, system, or repo, all scripts in `/var/lib/cobbler/triggers/add` are executed for the particular object type. Each particular file must be executable and it is executed with the name of the item being added as a parameter. Deletions work similarly – delete triggers live in `/var/lib/cobbler/triggers/delete`. Order of execution is arbitrary, and cobbler does not ship with any triggers by default. There are also other kinds of triggers – these are described on the Cobbler Wiki. For larger configurations, triggers should be written in Python – in which case they are installed differently. This is also documented on the Wiki.

6.5 Images

Cobbler can help with booting images physically and virtually, though the usage of these commands varies substantially by the type of image. Non-image based deployments are generally easier to work with and lead to more sustaintable infrastructure. Some manual use of other commands beyond of what is typically required of cobbler may be needed to prepare images for use with this feature.

6.6 Power Management

Cobbler contains a power management feature that allows the user to associate system records in cobbler with the power management configuration attached to them. This can ease installation by making it easy to reassign systems to new operating systems and then reboot those systems. Read more about this feature at https://github.com/cobbler/cobbler/wiki/Power-management

6.7 Non-import (manual) workflow

The following example uses a local kernel and initrd file (already downloaded), and shows how profiles would be created using two different automatic installation files – one for a web server configuration and one for a database server. Then, a machine is assigned to each profile.

```
cobbler check
cobbler distro add --name=rhel4u3 --kernel=/dir1/vmlinuz --initrd=/dir1/initrd.img
cobbler distro add --name=fc5 --kernel=/dir2/vmlinuz --initrd=/dir2/initrd.img
cobbler profile add --name=fc5webservers --distro=fc5-i386 --autoinst=/dir4/kick.
↪ks --kopts="something_to_make_my_gfx_card_work=42 some_other_parameter=foo"
cobbler profile add --name=rhel4u3dbservers --distro=rhel4u3 --autoinst=/dir5/kick.
↪ks
cobbler system add --name=AA:BB:CC:DD:EE:FF --profile=fc5-webservers
cobbler system add --name=AA:BB:CC:DD:EE:FE --profile=rhel4u3-dbservers
cobbler report
```

6.8 Repository Management

6.8.1 REPO MANAGEMENT

This has already been covered a good bit in the command reference section.

Yum repository management is an optional feature, and is not required to provision through cobbler. However, if cobbler is configured to mirror certain repositories, it can then be used to associate profiles with those repositories. Systems installed under those profiles will then be autoconfigured to use these repository mirrors in `/etc/yum.repos.d`, and if supported (Fedora Core 6 and later) these repositories can be leveraged even within Anaconda. This can be useful if (A) you have a large install base, (B) you want fast installation and upgrades for your

systems, or (C) have some extra software not in a standard repository but want provisioned systems to know about that repository.

Make sure there is plenty of space in cobbler's webdir, which defaults to `/var/www/cobbler`.

```
cobbler reposync [--tries=N] [--no-fail]
```

Cobbler reposync is the command to use to update repos as configured with "cobbler repo add". Mirroring can take a long time, and usage of cobbler reposync prior to usage is needed to ensure provisioned systems have the files they need to actually use the mirrored repositories. If you just add repos and never run "cobbler reposync", the repos will never be mirrored. This is probably a command you would want to put on a crontab, though the frequency of that crontab and where the output goes is left up to the systems administrator.

For those familiar with yum's reposync, cobbler's reposync is (in most uses) a wrapper around the yum command. Please use "cobbler reposync" to update cobbler mirrors, as yum's reposync does not perform all required steps. Also cobbler adds support for rsync and SSH locations, where as yum's reposync only supports what yum supports (http/ftp).

If you ever want to update a certain repository you can run:

```
cobbler reposync --only="reponame1" ...
```

When updating repos by name, a repo will be updated even if it is set to be not updated during a regular reposync operation (ex: cobbler repo edit –name=reponame1 –keep-updated=0).

Note that if a cobbler import provides enough information to use the boot server as a yum mirror for core packages, cobbler can set up automatic installation files to use the cobbler server as a mirror instead of the outside world. If this feature is desirable, it can be turned on by setting yum_post_install_mirror to 1 in /etc/settings (and running "cobbler sync"). You should not use this feature if machines are provisioned on a different VLAN/network than production, or if you are provisioning laptops that will want to acquire updates on multiple networks.

The flags `--tries=N` (for example, `--tries=3`) and `--no-fail` should likely be used when putting reposync on a crontab. They ensure network glitches in one repo can be retried and also that a failure to synchronize one repo does not stop other repositories from being synchronized.

6.8.2 Importing trees

Cobbler can auto-add distributions and profiles from remote sources, whether this is a filesystem path or an rsync mirror. This can save a lot of time when setting up a new provisioning environment. Import is a feature that many users will want to take advantage of, and is very simple to use.

After an import is run, cobbler will try to detect the distribution type and automatically assign automatic installation files. By default, it will provision the system by erasing the hard drive, setting up eth0 for dhcp, and using a default password of "cobbler". If this is undesirable, edit the automatic installation files in `/etc/cobbler` to do something else or change the automatic installation setting after cobbler creates the profile.

Mirrored content is saved automatically in `/var/www/cobbler/distro_mirror`.

Example 1: `cobbler import --path=rsync://mirrorserver.example.com/path/ --name=fedora --arch=x86`

Example 2: `cobbler import --path=root@192.168.1.10:/stuff --name=bar`

Example 3: `cobbler import --path=/mnt/dvd --name=baz --arch=x86_64`

Example 4: `cobbler import --path=/path/to/stuff --name=glorp`

Example 5: `cobbler import --path=/path/where/filer/is/mounted --name=anyname --available-as=nfs://nfs.example.org:/where/mounted/`

Once imported, run a `cobbler list` or `cobbler report` to see what you've added.

By default, the rsync operations will exclude content of certain architectures, debug RPMs, and ISO images – to change what is excluded during an import, see `/etc/cobbler/rsync.exclude`.

Note that all of the import commands will mirror install tree content into `/var/www/cobbler` unless a network accessible location is given with `--available-as`. –available-as will be primarily used when importing distros stored on an external NAS box, or potentially on another partition on the same machine that is already accessible via `http://` or `ftp://`.

For import methods using rsync, additional flags can be passed to rsync with the option `--rsync-flags`.

Should you want to force the usage of a specific cobbler automatic installation template for all profiles created by an import, you can feed the option `--autoinst` to import, to bypass the built-in automatic installation file auto-detection.

6.8.3 Repository mirroring workflow

The following example shows how to set up a repo mirror for two repositories, and create a profile that will auto install those repository configurations on provisioned systems using that profile.

```
cobbler check
# set up your cobbler distros here.
cobbler repo add --mirror=http://mirrors.kernel.org/fedora/core/updates/6/i386/ --
↪name=fc6i386updates
cobbler repo add --mirror=http://mirrors.kernel.org/fedora/extras/6/i386/ --
↪name=fc6i386extras
cobbler reposync
cobbler profile add --name=p1 --distro=existing_distro_name --autoinst=/etc/
↪cobbler/kickstart_fc6.ks --repos="fc6i386updates fc6i386extras"
```

6.8.4 Import Workflow

Import is a very useful command that makes starting out with cobbler very quick and easy.

This example shows how to create a provisioning infrastructure from a distribution mirror or DVD ISO. Then a default PXE configuration is created, so that by default systems will PXE boot into a fully automated install process for that distribution.

You can use a network rsync mirror, a mounted DVD location, or a tree you have available via a network filesystem.

Import knows how to autodetect the architecture of what is being imported, though to make sure things are named correctly, it's always a good idea to specify `--arch`. For instance, if you import a distribution named "fedora8" from an ISO, and it's an x86_64 ISO, specify `--arch=x86_64` and the distro will be named "fedora8-x86_64" automatically, and the right architecture field will also be set on the distribution object. If you are batch importing an entire mirror (containing multiple distributions and arches), you don't have to do this, as cobbler will set the names for things based on the paths it finds.

```
cobbler check
cobbler import --path=rsync://yourfavoritemirror.com/rhel/5/os/x86_64 --name=rhel5␣
↪--arch=x86_64
# OR
cobbler import --path=/mnt/dvd --name=rhel5 --arch=x86_64
# OR (using an external NAS box without mirroring)
cobbler import --path=/path/where/filer/is/mounted --name=anyname --available-
↪as=nfs://nfs.example.org:/where/mounted/
# wait for mirror to rsync...
cobbler report
cobbler system add --name=default --profile=name_of_a_profile1
cobbler system add --name=AA:BB:CC:DD:EE:FF --profile=name_of_a_profile2
cobbler sync
```

6.9 Virtualization

For Virt, be sure the distro uses the correct kernel (if paravirt) and follow similar steps as above, adding additional parameters as desired:

```
cobbler distro add --name=fc7virt [options...]
```

Specify reasonable values for the Virt image size (in GB) and RAM requirements (in MB):

```
cobbler profile add --name=virtwebservers --distro=fc7virt --autoinst=path --virt-
↪file-size=10 --virt-ram=512 [...]
```

Define systems if desired. koan can also provision based on the profile name.

```
cobbler system add --name=AA:BB:CC:DD:EE:FE --profile=virtwebservers [...]
```

If you have just installed cobbler, be sure that the "cobblerd" service is running and that port 25151 is unblocked.

See the manpage for koan for the client side steps.

6.10 Autoinstallation

6.10.1 Automatic installation templating

The `--autoinstall_meta` options above require more explanation.

If and only if `--autoinst` options reference filesystem URLs, `--ksmeta` allows for templating of the automatic installation files to achieve advanced functions. If the `--ksmeta` option for a profile read `--ksmeta="foo=7 bar=llama"`, anywhere in the automatic installation file where the string `$bar` appeared would be replaced with the string "llama".

To apply these changes, `cobbler sync` must be run to generate custom automatic installation files for each profile/system.

For NFS and HTTP automatic installation file URLs, the `--autoinstall_meta` options will have no effect. This is a good reason to let cobbler manage your automatic installation files, though the URL functionality is provided for integration with legacy infrastructure, possibly including web apps that already generate automatic installation files.

Templated automatic files are processed by the templating program/package Cheetah, so anything you can do in a Cheetah template can be done to an automatic installation template. Learn more at http://www.cheetahtemplate.org/learn.html

When working with Cheetah, be sure to escape any shell macros that look like `$(this)` with something like `\$(this)` or errors may show up during the sync process.

The Cobbler Wiki also contains numerous Cheetah examples that should prove useful in using this feature.

Also usefull ist the following repo: https://github.com/FlossWare/cobbler

6.10.2 Automatic installation snippets

Anywhere a automatic installation template mentions `SNIPPET::snippet_name`, the file named `/var/lib/cobbler/snippets/snippet_name` (if present) will be included automatically in the automatic installation template. This serves as a way to recycle frequently used automatic installation snippets without duplication. Snippets can contain templating variables, and the variables will be evaluated according to the profile and/or system as one would expect.

Snippets can also be overridden for specific profile names or system names. This is described on the Cobbler Wiki.

6.10.3 Kickstart validation

To check for potential errors in kickstarts, prior to installation, use `cobbler validateks`. This function will check all profile and system kickstarts for detectable errors. Since pykickstart is not future-Anaconda-version aware, there may be some false positives. It should be noted that `cobbler validateks` runs on the rendered kickstart output, not kickstart templates themselves.

6.11 Network Topics

6.11.1 PXE Menus

Cobbler will automatically generate PXE menus for all profiles it has defined. Running `cobbler sync` is required to generate and update these menus.

To access the menus, type `menu` at the `boot:` prompt while a system is PXE booting. If nothing is typed, the network boot will default to a local boot. If "menu" is typed, the user can then choose and provision any cobbler profile the system knows about.

If the association between a system (MAC address) and a profile is already known, it may be more useful to just use `system add` commands and declare that relationship in cobbler; however many use cases will prefer having a PXE system, especially when provisioning is done at the same time as installing new physical machines.

If this behavior is not desired, run `cobbler system add --name=default --profile=plugh` to default all PXE booting machines to get a new copy of the profile `plugh`. To go back to the menu system, run `cobbler system remove --name=default` and then `cobbler sync` to regenerate the menus.

When using PXE menu deployment exclusively, it is not necessary to make cobbler system records, although the two can easily be mixed.

Additionally, note that all files generated for the pxe menu configurations are templatable, so if you wish to change the color scheme or equivalent, see the files in `/etc/cobbler`.

6.11.2 Default PXE Boot behavior

What happens when PXE booting a system when cobbler has no record of the system being booted?

By default, cobbler will configure PXE to boot to the contents of `/etc/cobbler/default.pxe`, which (if unmodified) will just fall through to the local boot process. Administrators can modify this file if they like to change that behavior.

An easy way to specify a default cobbler profile to PXE boot is to create a system named `default`. This will cause `/etc/cobbler/default.pxe` to be ignored. To restore the previous behavior do a `cobbler system remove` on the `default` system.

```
cobbler system add --name=default --profile=boot_this
cobbler system remove --name=default
```

As mentioned in earlier sections, it is also possible to control the default behavior for a specific network:

```
cobbler system add --name=network1 --ip-address=192.168.0.0/24 --profile=boot_this
```

6.11.3 PXE boot loop prevention

If you have your machines set to PXE first in the boot order (ahead of hard drives), change the `pxe_just_once` flag in `/etc/cobbler/settings` to 1. This will set the machines to not PXE on successive boots once they complete one install. To re-enable PXE for a specific system, run the following command:

```
cobbler system edit --name=name --netboot-enabled=1
```

6.11.4 Automatic installation tracking

Cobbler knows how to keep track of the status of automatic installation of machines.

```
cobbler status
```

Using the status command will show when cobbler thinks a machine started automatic installation and when it finished, provided the proper snippets are found in the automatic installation template. This is a good way to track machines that may have gone interactive (or stalled/crashed) during automatic installation.

6.12 Boot CD

Cobbler can build all of it's profiles into a bootable CD image using the `cobbler buildiso` command. This allows for PXE-menu like bringup of bare metal in environments where PXE is not possible. Another more advanced method is described in the koan manpage, though this method is easier and sufficient for most applications.

6.12.1 DHCP Management

Cobbler can optionally help you manage DHCP server. This feature is off by default.

Choose either `management = isc_and_bind` in `/etc/cobbler/dhcp.template` or `management = "dnsmasq"` in `/etc/cobbler/modules.conf`. Then set `manage_dhcp=1` in `/etc/cobbler/settings`.

This allows DHCP to be managed via "cobbler system add" commands, when you specify the mac address and IP address for systems you add into cobbler.

Depending on your choice, cobbler will use `/etc/cobbler/dhcpd.template` or `/etc/cobbler/dnsmasq.template` as a starting point. This file must be user edited for the user's particular networking environment. Read the file and understand how the particular app (ISC dhcpd or dnsmasq) work before proceeding.

If you already have DHCP configuration data that you would like to preserve (say DHCP was manually configured earlier), insert the relevant portions of it into the template file, as running `cobbler sync` will overwrite your previous configuration.

By default, the DHCP configuration file will be updated each time `cobbler sync` is run, and not until then, so it is important to remember to use `cobbler sync` when using this feature.

If omapi_enabled is set to 1 in `/etc/cobbler/settings`, the need to sync when adding new system records can be eliminated. However, the omapi feature is experimental and is not recommended for most users.

6.12.2 DNS configuration management

Cobbler can optionally manage DNS configuration using BIND and dnsmasq.

Choose either `management = isc_and_bind` or `management = dnsmasq` in `/etc/cobbler/modules.conf` and then enable `manage_dns` in `/etc/cobbler/settings`.

This feature is off by default. If using BIND, you must define the zones to be managed with the options `manage_forward_zones` and `manage_reverse_zones`. (See the Wiki for more information on this).

If using BIND, Cobbler will use `/etc/cobbler/named.template` and `/etc/cobbler/zone.template` as a starting point for the `named.conf` and individual zone files, respectively. You may drop zone-specific template files in `/etc/cobbler/zone_templates/name-of-zone` which will override the

default. These files must be user edited for the user's particular networking environment. Read the file and understand how BIND works before proceeding.

If using dnsmasq, the template is `/etc/cobbler/dnsmasq.template`. Read this file and understand how dnsmasq works before proceeding.

All managed files (whether zone files and `named.conf` for BIND, or `dnsmasq.conf` for dnsmasq) will be updated each time `cobbler sync` is run, and not until then, so it is important to remember to use `cobbler sync` when using this feature.

6.13 Containerization

We have a test-image which you can find in the cobbler repository and an old image made by the community: https://github.com/osism/docker-cobbler

CHAPTER 7

Developer Guide

7.1 Patch process

You'd like to contribute features or fixes to Cobbler? Great! We'd love to have them.

It is highly recommended that you have a github.com account if you would like to contribute code. Create an account, log in, and then go to github.com/cobbler/cobbler to "fork" the project.

Create a new branch named after the feature you are working on. Do the work on your local machine, please make sure your work passes Cobbler's coding standards by using `make qa`. Only then push to your personal Github branch (e.g. github.com/yourname/cobbler).

Then use the "submit pull request" feature of Github to request that the official repo pull in your changes. Be sure to include a full description of what your change does in the comments, including what you have tested (and other things that you may have not been able to test well and need help with).

If the patch needs more work, we'll let you know in the comments.

Do not mix work on different features in different pull requests/branches if at all possible as this makes it difficult to take only some of the work at one time, and to quickly slurp in some changes why others get hammered out.

Once we merge in your pull request, you can remove the branch from your repo if you like. The AUTHORS file is created automatically when we release.

7.2 Setup

The preferred development platform is the latest openSUSE Leap or Tumbleweed. You'll also have to disable SELinux to get Cobbler up and running.

For CentOS you will need the EPEL repository: `http://download.fedoraproject.org/pub/epel/7/x86_64/repoview/epel-release.html`

Install development dependencies:

```
# yum install git make openssl python-sphinx python36-coverage python36-devel␣
↪python36-distro python36-future python36-pyflakes python36-pycodestyle python36-
↪setuptools rpm-build
```

Install runtime dependencies:

```
# yum install httpd mod_wsgi python36-PyYAML python36-netaddr python36-simplejso m
# pip3 install Cheetah3
```

Initially, to run Cobbler without using packages:

```
# git clone https://github.com/<your username>/cobbler.git
# cd cobbler
# make install
```

For each successive run, do not run make install again. To avoid blowing away your configuration, run:

```
# make webtest
```

This will install Cobbler and restart apache/cobblerd, but move your configuration files and settings aside and restore them, rather than blindly overwriting them.

You can now run Cobbler commands and access the web interface.

7.3 Branches

Cobbler has a development branch called "master" (where the action is), and branches for all releases that are in maintaince mode. All work on new features should be done against the master branch. If you want to address bugs then please target the latest release branch, the maintainers will then cherry-pick those changes into the master branch.

```
# git branch -r
# git checkout <branch>
# git checkout -b <new branch name>
```

7.4 Standards

We're not overly picky, but please follow the python PEP8 standards we want to adhere to (see Makefile).

- Always use under_scores, not camelCase.
- Always four (4) spaces, not tabs.
- Avoid one line if statements.
- Validate your code by using `make qa`.
- Keep things simple, keep in mind that this is a tool for sysadmins and not python developers.
- Use modules that are easily available (eg. EPEL) but preferrably in the base OS, otherwise they have to be packaged with the app, which usually runs afoul of distribution packaging guidelines.
- Cobbler is since the 3.x.x release Python3 only.
- Koan has no new release currently but starting with the next we will also only support Pyhton3.
- Older releases will of course stay with Python2.

You're also welcome to hang out in #cobbler and #cobbler-devel on irc.freenode.net, as there are folks around to answer questions, etc. But it isn't that active anymore please drop also in our Cobbler Gitter channel there we will probably answer faster.

7.5 Contributing to the website

The github-based git repository for the https://cobbler.github.io website itself is at https://github.com/cobbler/cobbler.github.io.

If you want to contribute changes to the website, you will need jekyll (http://jekyllrb.com).

You will probably want to:

- edit the files as markdown
- run the docker container
- check if your changes didn't break anything

7.6 Debugging

If you need to debug a remote process, rpdb provides some very nice capabilities beyond the standard python debugger, just insert a *import rpdb; rpdb.set_trace()* on the desired line run cobbler and then do a *nc 127.0.0.1 4444*.

CHAPTER 8

cobbler package

8.1 Subpackages

8.1.1 cobbler.actions package

Submodules

cobbler.actions.acl module

Configures acls for various users/groups so they can access the Cobbler command line as non-root. Now that CLI is largely remoted (XMLRPC) this is largely just useful for not having to log in (access to shared-secret) file but also grants access to hand-edit various cobbler_collections files and other useful things.

class cobbler.actions.acl.**AclConfig**(*collection_mgr*, *logger=None*)

Bases: object

modacl(*isadd*, *isuser*, *who*)

Modify the acls for Cobbler on the filesystem.

Parameters

- **isadd** (*bool*) – If true then the who will be added. If false then who will be removed.
- **isuser** (*bool*) – If true then the who may be a user. If false then who may be a group.
- **who** – The user or group to be added or removed.

run (*adduser=None*, *addgroup=None*, *removeuser=None*, *removegroup=None*)
Automate setfacl commands. Only one of the four may be specified but one option also must be specified.

Parameters

- **adduser** – Add a user to be able to manage Cobbler.
- **addgroup** – Add a group to be able to manage Cobbler.
- **removeuser** – Remove a user to be able to manage Cobbler.
- **removegroup** – Remove a group to be able to manage Cobbler.

cobbler.actions.buildiso module

Builds bootable CD images that have PXE-equivalent behavior for all Cobbler distros/profiles/systems currently in memory.

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.actions.buildiso.**BuildIso**(*collection_mgr*, *verbose=False*, *logger=None*)
Bases: object

Handles conversion of internal state to the isolinux tree layout

add_remaining_kopts(*koptdict*)
Add remaining kernel_options to append_line

Parameters **koptdict** (*dict*) – The kernel options which are not present in append_line.

Returns A single line with all kernel options

Return type str

copy_boot_files(*distro*, *destdir*, *prefix=None*)
Copy kernel/initrd to destdir with (optional) newfile prefix

Parameters

- **distro** – Distro object to return the boot files for.
- **destdir** – The destionation direcotry.
- **prefix** (*str*) – The file prefix.

filter_systems_or_profiles(*selected_items*, *list_type*)
Return a list of valid profile or system objects selected from all profiles or systems by name, or everything if selected_items is empty.

Parameters

- **selected_items** – The filter to match certain objects with. The filter will be applied to the object name.
- **list_type** (*str*) – Must be "profile" or "system".

Returns A list of valid profiles OR systems.

Return type list

generate_netboot_iso(*imagesdir*, *isolinuxdir*, *profiles=None*, *systems=None*, *exclude_dns=None*)
Create bootable CD image to be used for network installations

Parameters

- **imagesdir** – Currently unused parameter.
- **isolinuxdir** – The parent directory where the isolinux.cfg is located.
- **profiles** – The filter to generate a netboot iso for. You may specify multiple profiles on the CLI space separated.
- **systems** – The filter to generate a netboot iso for. You may specify multiple systems on the CLI space separated.
- **exclude_dns** (*bool or None*) – If this is True then the dns server is skipped. None or False will set it.

generate_standalone_iso(*imagesdir*, *isolinuxdir*, *distname*, *filesource*, *airgapped*, *profiles*)
Create bootable CD image to be used for handsoff CD installtions

Parameters

- **imagesdir** – Unused Parameter.
- **isolinuxdir** – The parent directory where the file isolinux.cfg is located at.
- **distname** – The name of the Cobbler distribution.
- **filesource** – Not clear what this exactly does
- **airgapped** (*bool*) – Whether the repositories have to be locally available or the internet is reachable.
- **profiles** – The list of profiles to include.

make_shorter(*distname*)
Return A short distro identifier.

Parameters **distname** (*str*) – The distro name to return a identifieer for.

Returns A short distro identifier

Return type str

run(*iso=None*, *buildisodir=None*, *profiles=None*, *systems=None*, *distro=None*, *standalone=None*, *airgapped=None*, *source=None*, *exclude_dns=None*, *xorrisofs_opts=None*)
A

Parameters

- **iso** – The name of the iso. Defaults to "autoinst.iso".
- **buildisodir** – This overwrites the directory from the settings in which the iso is built in.
- **profiles** –
- **systems** – Don't use that when building standalone isos.
- **distro** – (For standalone only)
- **standalone** (*bool*) – This means that no network connection is needed to install the generated iso.
- **airgapped** (*bool*) – This option implies standalone=True.
- **source** – If the iso should be offline available this is the path to the sources of the image.

- **exclude_dns** – Whether the repositories have to be locally available or the internet is reachable.
- **xorrisofs_opts** – xorrisofs options to include additionally.

cobbler.actions.check module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.actions.check.**CobblerCheck**(*collection_mgr*, *logger=None*)
Bases: object

Validates whether the system is reasonably well configured for serving up content. This is the code behind 'cobbler check'.

check_bind_bin(*status*)
Check if bind is installed.

Parameters **status** – The status list with possible problems.

check_bootloaders(*status*)
Check if network bootloaders are installed

Parameters **status** – The status list with possible problems.

check_ctftpd_dir(*status*)
Check if cobbler.conf's tftpboot directory exists.

Parameters **status** – The status list with possible problems.

check_debmirror(*status*)
Check if debmirror is available and the config file for it exists. If the distro familiy is suse then this will pass without checking.

Parameters **status** – The status list with possible problems.

check_dhcpd_bin(*status*)
Check if dhcpd is installed.

Parameters **status** – The status list with possible problems.

check_dhcpd_conf(*status*)
NOTE: this code only applies if Cobbler is *NOT* set to generate a dhcp.conf file.

Check that dhcpd *appears* to be configured for pxe booting. We can't assure file correctness. Since a Cobbler user might have dhcp on another server, it's okay if it's not there and/or not configured correctly according to automated scans.

Parameters **status** – The status list with possible problems.

check_dnsmasq_bin(*status*)
Check if dnsmasq is installed.

Parameters **status** – The status list with possible problems.

check_for_cman(*status*)
Check if the fencing tools are available. This is done thorugh checking if the binary `fence_ilo` is present in `/sbin` or `/usr/sbin`.

Parameters **status** – The status list with possible problems. The status list with possible problems.

check_for_default_password(*status*)
Check if the default password of Cobbler was changed.

Parameters **status** – The status list with possible problems.

check_for_ksvalidator(*status*)
Check if the `ksvalidator` is present in `/usr/bin`.

Parameters **status** – The status list with possible problems. The status list with possible problems.

check_for_unreferenced_repos(*status*)
Check if there are repositories which are not used and thus could be removed.

Parameters **status** – The status list with possible problems.

check_for_unsynced_repos(*status*)
Check if there are unsynchronized repositories which need an update.

Parameters **status** – The status list with possible problems.

check_for_wget_curl(*status*)
Check to make sure wget or curl is installed

Parameters **status** – The status list with possible problems.

check_iptables(*status*)
Check if iptables is running. If yes print the needed ports. This is unavailable on Debian, SUSE and CentOS7 as a service. However this only indicates that the way of persisting the iptable rules are persisted via other means.

Parameters **status** – The status list with possible problems.

check_name(*status*)
If the server name in the config file is still set to localhost automatic installations run from koan will not have proper kernel line parameters.

Parameters **status** – The status list with possible problems.

check_rsync_conf(*status*)
Check that rsync is enabled to autostart.

Parameters **status** – The status list with possible problems.

check_selinux(*status*)
Suggests various SELinux rules changes to run Cobbler happily with SELinux in enforcing mode.

Parameters **status** – The status list with possible problems.

check_service(*status*, *which*, *notes=''*)
Check if the service command is available or the old init.d system has to be used.

Parameters

- **status** – The status list with possible problems.
- **which** – The service to check for.
- **notes** – A manual not to attach.

check_tftpd_dir(*status*)
Check if cobbler.conf's tftpboot directory exists

Parameters **status** – The status list with possible problems.

check_yum(*status*)
Check if the yum-stack is available. On Debian based distros this will always return without checking.

Parameters **status** – The status list with possible problems.

run()
The CLI usage is "cobbler check" before "cobbler sync".

Returns None if there are no errors, otherwise returns a list of things to correct prior to running application 'for real'.

cobbler.actions.dlcontent module

Downloads bootloader content for all arches for when the user doesn't want to supply their own.

Copyright 2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.actions.dlcontent.**ContentDownloader**(*collection_mgr*, *logger=None*)
Bases: object

run(*force=False*)
Download bootloader content for all of the latest bootloaders, since the user has chosen to not supply their own. You may ask "why not get this from yum", we also want this to be able to work on Debian and further do not want folks to have to install a cross compiler. For those that don't like this approach they can still source their cross-arch bootloader content manually.

Parameters **force** (*bool*) – If the target path should be overwritten, even if there are already files present.

cobbler.actions.hardlink module

Hard links Cobbler content together to save space.

Copyright 2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.actions.hardlink.**HardLinker**(*collection_mgr*, *logger=None*)
Bases: object

run()
Simply hardlinks directories that are Cobbler managed. This is a /very/ simple command but may grow more complex and intelligent over time.

cobbler.actions.litesync module

Running small pieces of Cobbler sync when certain actions are taken, such that we don't need a time consuming sync when adding new systems if nothing has changed for systems that have already been created.

class cobbler.actions.litesync.**CobblerLiteSync**(*collection_mgr*, *verbose=False*, *logger=None*)

Bases: object

Handles conversion of internal state to the tftpboot tree layout

add_single_distro(*name*)
Sync adding a single distro.

Parameters **name** – The name of the distribution.

add_single_image(*name*)
Sync adding a single image.

Parameters **name** – The name of the image.

add_single_profile(*name*, *rebuild_menu=True*)
Sync adding a single profile.

Parameters

- **name** (*str*) – The name of the profile.
- **rebuild_menu** (*bool*) – Whether to rebuild the grub/... menu or not.

Returns True if this succeeded.

add_single_system(*name*)
Sync adding a single system.

Parameters **name** (*str*) – The name of the system.

remove_single_distro(*name*)
Sync removing a single distro.

Parameters **name** – The name of the distribution.

remove_single_image(*name*)
Sync removing a single image.

Parameters **name** – The name of the image.

remove_single_profile(*name*, *rebuild_menu=True*)
Sync removing a single profile.

Parameters

- **name** (*str*) – The name of the profile.
- **rebuild_menu** (*bool*) – Whether to rebuild the grub/... menu or not.

remove_single_system(*name*)
Sync removing a single system.

Parameters **name** (*str*) – The name of the system.

update_system_netboot_status(*name*)
Update the netboot status of a system.

Parameters **name** (*str*) – The name of the system.

cobbler.actions.log module

Copyright 2009, Red Hat, Inc and Others Bill Peck <bpeck@redhat.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.actions.log.**LogTool**(*collection_mgr*, *system*, *api*, *logger=None*)
Bases: object

Helpers for dealing with System logs, anamon, etc..

clear()
Clears the system logs

cobbler.actions.replicate module

Replicate from a Cobbler master.

Copyright 2007-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail> Scott Henson <shenson@redhat.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.actions.replicate.**Replicate**(*collection_mgr*, *logger=None*)
Bases: object

add_objects_not_on_local(*obj_type*)
Add objects locally which are not present on the slave but on the master.

Parameters **obj_type** –

generate_include_map()
Not known what this exactly does.

link_distros()
Link a distro from its location into the web directory to make it available for usage.

remove_objects_not_on_master(*obj_type*)
Remove objects on this slave which are not on the master.

Parameters **obj_type** – The type of object which should be synchronized.

replace_objects_newer_on_remote(*obj_type*)
Replace objects which are newer on the local slave then on the remote slave

Parameters **obj_type** – The type of object to synchronize.

replicate_data()
Replicate the local and remote data to each another.

rsync_it(*from_path*, *to_path*, *type=None*)
Rsync from a source to a destination with the rsync options Cobbler was configured with.

Parameters

- **from_path** – The source to rsync from.
- **to_path** – The destination to rsync to.
- **type** – If set to "repo" this will take the repo rsync options instead of the global ones.

run(*cobbler_master=None*, *port='80'*, *distro_patterns=None*, *profile_patterns=None*, *system_patterns=None*, *repo_patterns=None*, *image_patterns=None*, *mgmtclass_patterns=None*, *package_patterns=None*, *file_patterns=None*, *prune=False*, *omit_data=False*, *sync_all=False*, *use_ssl=False*)
Get remote profiles and distros and sync them locally

Parameters

- **cobbler_master** – The remote url of the master server.
- **port** – The remote port of the master server.
- **distro_patterns** – The pattern of distros to sync.
- **profile_patterns** – The pattern of profiles to sync.
- **system_patterns** – The pattern of systems to sync.
- **repo_patterns** – The pattern of repositories to sync.
- **image_patterns** – The pattern of images to sync.
- **mgmtclass_patterns** – The pattern of management classes to sync.
- **package_patterns** – The pattern of packages to sync.
- **file_patterns** – The pattern of files to sync.
- **prune** – If the local server should be pruned before coping stuff.
- **omit_data** – If the data behind images etc should be omitted or not.
- **sync_all** – If everything should be synced (then the patterns are useless) or not.
- **use_ssl** – If HTTPS or HTTP should be used.

cobbler.actions.report module

Report from a Cobbler master. FIXME: reinstante functionality for 2.0

Copyright 2007-2009, Red Hat, Inc and Others Anderson Silva <ansilva@redhat.com> Michael DeHaan <michael.dehaan AT gmail>

This software may be freely redistributed under the terms of the GNU general public license.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

class cobbler.actions.report.**Report**(*collection_mgr*, *logger=None*)
Bases: object

fielder(*structure*, *fields_list*)
Return data from a subset of fields of some item

Parameters

- **structure** – The item structure to report.
- **fields_list** – The list of fields which should be returned.

Returns The same item with only the given subset of information.

print_formatted_data(*data*, *order*, *report_type*, *noheaders*)
Used for picking the correct format to output data as

Parameters

- **data** – The list of iteratable items for table output.
- **order** – The list of fields which are available in the table file.
- **noheaders** – Whether headers are printed to the output or not.
- **report_type** – The type of report which should be used.

reporting_csv(*info*, *order*, *noheaders*)
Formats data on 'info' for csv output

Parameters

- **info** – The list of iteratable items for csv output.
- **order** – The list of fields which are available in the csv file.
- **noheaders** – Whether headers are printed to the output or not.

Returns The string with the csv.

reporting_doku(*info*, *order*, *noheaders*)
Formats data on 'info' for doku wiki table output

Parameters

- **info** – The list of iteratable items for table output.
- **order** – The list of fields which are available in the table file.
- **noheaders** – Whether headers are printed to the output or not.

Returns The string with the generated table.

reporting_list_names2(*collection*, *name*)
Prints a specific object in a collection.

Parameters

- **collection** – The collections object to print a collection from.
- **name** – The name of the collection to print.

reporting_mediawiki(*info*, *order*, *noheaders*)
Formats data on 'info' for mediawiki table output

Parameters

- **info** – The list of iteratable items for table output.
- **order** – The list of fields which are available in the table file.
- **noheaders** – Whether headers are printed to the output or not.

Returns The string with the generated table.

reporting_print_all_fields(*collection*, *report_name*, *report_type*, *report_noheaders*)
Prints all fields in a collection as a table given the report type

Parameters

- **collection** – The collection to report.

- **report_name** – The name of the report.
- **report_type** – The type of report to give.
- **report_noheaders** – Report without the headers. (May be useful for machine parsing)

Returns A report with all fields included pretty printed or machine readable.

reporting_print_sorted(*collection*)
Prints all objects in a collection sorted by name

Parameters **collection** – The collection to print.

reporting_print_x_fields(*collection*, *report_name*, *report_type*, *report_fields*, *report_noheaders*)
Prints specific fields in a collection as a table given the report type

Parameters

- **collection** – The collection to report.
- **report_name** – The name of the report.
- **report_type** – The type of report to give.
- **report_fields** – The fields which should be included in the report.
- **report_noheaders** – Report without the headers. (May be useful for machine parsing)

reporting_sorter(*a*, *b*)
Used for sorting Cobbler objects for report commands

Parameters

- **a** – The first object to compare.
- **b** – The second object to compare.

Returns Whether the first or second object is greater or smaller.

reporting_trac(*info*, *order*, *noheaders*)
Formats data on 'info' for trac wiki table output

Parameters

- **info** – The list of iteratable items for table output.
- **order** – The list of fields which are available in the table file.
- **noheaders** – Whether headers are printed to the output or not.

Returns The string with the generated table.

run(*report_what=None*, *report_name=None*, *report_type=None*, *report_fields=None*, *report_noheaders=None*)
Get remote profiles and distros and sync them locally

1. Handles original report output
2. Handles all fields of report outputs as table given a format
3. Handles specific fields of report outputs as table given a format

Parameters

- **report_what** – What should be reported. May be "all".
- **report_name** – The name of the report.
- **report_type** – The type of report to give.
- **report_fields** – The fields which should be included in the report.

- **report_noheaders** – Report without the headers. (May be useful for machine parsing)

cobbler.actions.reposync module

Builds out and synchronizes yum repo mirrors. Initial support for rsync, perhaps reposync coming later.

class cobbler.actions.reposync.**RepoSync**(*collection_mgr*, *tries=1*, *nofail=False*, *logger=None*)

Bases: object

Handles conversion of internal state to the tftpboot tree layout.

apt_sync(*repo*)

Handle copying of http:// and ftp:// debian repos.

Parameters **repo** – The apt repository to sync.

create_local_file(*dest_path*, *repo*, *output=True*)

Creates Yum config files for use by reposync

Two uses: (A) output=True, Create local files that can be used with yum on provisioned clients to make use of this mirror. (B) output=False, Create a temporary file for yum to feed into yum for mirroring

Parameters

- **dest_path** (*str*) – The destination path to create the file at.
- **repo** – The repository object to create a file for.
- **output** (*bool*) – See described above.

Returns The name of the file which was written.

createrepo_walker(*repo*, *dirname*, *fnames*)

Used to run createrepo on a copied Yum mirror.

Parameters

- **repo** – The repository object to run for.
- **dirname** – The directory to run in.
- **fnames** – Not known what this is for.

gen_urlgrab_ssl_opts(*yumopts*)

This function translates yum repository options into the appropriate options for python-requests

Parameters **yumopts** – The options to convert.

Returns A tuple with the cert and a boolean if it should be verified or not.

Return type (str, bool)

reposync_cmd()

Determine reposync command

Returns The path to the reposync command. If dnf exists it is used instead of reposync.

rhn_sync(*repo*)
Handle mirroring of RHN repos.

Parameters **repo** – The repo object to synchronize.

rsync_sync(*repo*)
Handle copying of rsync:// and rsync-over-ssh repos.

Parameters **repo** – The repo to sync via rsync.

run(*name=None*, *verbose=True*)
Syncs the current repo configuration file with the filesystem.

Parameters

- **name** – The name of the repository to synchronize.
- **verbose** (*bool*) – If the action should be logged verbose or not.

sync(*repo*)
Conditionally sync a repo, based on type.

Parameters **repo** – The repo to sync.

update_permissions(*repo_path*)
Verifies that permissions and contexts after an rsync are as expected. Sending proper rsync flags should prevent the need for this, though this is largely a safeguard.

Parameters **repo_path** – The path to update the permissions of.

wget_sync(*repo*)
Handle mirroring of directories using wget

Parameters **repo** – The repo object to sync via wget.

yum_sync(*repo*)
Handle copying of http:// and ftp:// yum repos.

Parameters **repo** – The yum reporitory to sync.

cobbler.actions.reposync.**repo_walker**(*top*, *func*, *arg*)
Directory tree walk with callback function.

For each directory in the directory tree rooted at top (including top itself, but excluding '.' and '..'), call func(arg, dirname, fnames). dirname is the name of the directory, and fnames a list of the names of the files and subdirectories in dirname (excluding '.' and '..'). func may modify the fnames list in-place (e.g. via del or slice assignment), and walk will only recurse into the subdirectories whose names remain in fnames; this can be used to implement a filter, or to impose a specific order of visiting. No semantics are defined for, or required of, arg, beyond that arg is always passed to func. It can be used, e.g., to pass a filename pattern, or a mutable object designed to accumulate statistics. Passing None for arg is common.

Parameters

- **top** – The directory that should be taken as root. The root dir will also be included in the processing.
- **func** – The function that should be executed.
- **arg** – The arguments for that function.

cobbler.actions.status module

Reports on automatic installation activity by examining the logs in /var/log/cobbler.

Copyright 2007-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.actions.status.**CobblerStatusReport**(*collection_mgr*, *mode*, *logger=None*)

Bases: object

catalog(*profile_or_system*, *name*, *ip*, *start_or_stop*, *ts*)

Add a system to cobbler status.

Parameters

- **profile_or_system** (*str*) – This can be system or profile.
- **name** (*str*) – The name of the object.
- **ip** – The ip of the system to watch.
- **start_or_stop** (*str*) – This parameter may be start or stop
- **ts** – Don't know what this does.

get_printable_results()

Convert the status of Cobbler from a machine readable form to human readable.

Returns A nice formatted representation of the results of cobbler status.

process_results()

Look through all systems which were collected and update the status.

Returns Return ip_data of the object.

run()

Calculate and print a automatic installation status report.

scan_logfiles()

Scan the install log-files - starting with the oldest file.

cobbler.actions.sync module

Builds out filesystem trees/data based on the object tree. This is the code behind 'cobbler sync'.

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.actions.sync.**CobblerSync**(*collection_mgr*, *verbose=True*, *dhcp=None*, *dns=None*, *logger=None*, *tftpd=None*)

Bases: object

Handles conversion of internal state to the tftpboot tree layout

clean_link_cache()

All files which are linked into the cache will be deleted so the cache can be rebuild.

clean_trees()
Delete any previously built pxelinux.cfg tree and virt tree info and then create directories.

Note: for SELinux reasons, some information goes in `/tftpboot`, some in `/var/www/cobbler` and some must be duplicated in both. This is because PXE needs tftp, and automatic installation and Virt operations need http. Only the kernel and initrd images are duplicated, which is unfortunate, though SELinux won't let me give them two contexts, so symlinks are not a solution. *Otherwise* duplication is minimal.

make_tftpboot()
Make directories for tftpboot images

rsync_gen()
Generate rsync modules of all repositories and distributions

run()
Syncs the current configuration file with the config tree. Using the `Check().run_` functions previously is recommended

sync_dhcp()
This calls write_dhcp and restarts the DHCP server.

write_dhcp()
Write all files which are associated to DHCP.

Module contents

8.1.2 cobbler.cobbler_collections package

Submodules

cobbler.cobbler_collections.collection module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.cobbler_collections.collection.**Collection**(*collection_mgr*)
Bases: `object`

Base class for any serializable list of things.

SEARCH_REKEY = {'dhcp-tag': 'dhcp_tag', 'inherit': 'parent', 'ip': 'ip_address',

add(*ref*, *save=False*, *with_copy=False*, *with_triggers=True*, *with_sync=True*, *quick_pxe_update=False*, *check_for_duplicate_names=False*, *check_for_duplicate_netinfo=False*, *logger=None*)
Add an object to the collection

Parameters

- **ref** – The reference to the object.
- **save** – If this is true then the objet is persited on the disk.

- **with_copy** – Is a bit of a misnomer, but lots of internal add operations can run with "with_copy" as False. True means a real final commit, as if entered from the command line (or basically, by a user). With with_copy as False, the particular add call might just be being run during deserialization, in which case extra semantics around the add don't really apply. So, in that case, don't run any triggers and don't deal with any actual files.
- **with_sync** – If a sync should be triggered when the object is renamed.
- **with_triggers** – If triggers should be run when the object is renamed.
- **quick_pxe_update** – This decides if there should be run a quick or full update after the add was done.
- **check_for_duplicate_names** (*bool*) – If the name of an object should be unique or not.
- **check_for_duplicate_netinfo** (*bool*) – This checks for duplicate network information. This only has an effect on systems.
- **logger** – The logger to audit the action with.

static collection_type() → str
Returns the string key for the name of the collection (used by serializer etc)

static collection_types() → str
Returns the string key for the plural name of the collection (used by serializer)

copy(*ref*, *newname*, *logger=None*)
Copy an object with a new name into the same collection.

Parameters

- **ref** – The reference to the object which should be copied.
- **newname** – The new name for the copied object.
- **logger** – This parameter is unused in this implementation.

factory_produce(*collection_mgr*, *seed_data*)
Must override in subclass. Factory_produce returns an Item object from dict.

Parameters

- **collection_mgr** – The collection manager to resolve all information with.
- **seed_data** –

find(*name=None*, *return_list=False*, *no_errors=False*, ***kargs*)
Return first object in the collection that maches all item='value' pairs passed, else return None if no objects can be found. When return_list is set, can also return a list. Empty list would be returned instead of None in that case.

Parameters

- **name** (*str*) – The object name which should be found.
- **return_list** – If a list should be returned or the first match.
- **no_errors** – If errors which are possibly thrown while searching should be ignored or not.
- **kargs** (*dict*) – If name is present, this is optional, otherwise this dict needs to have at least a key with `name`. You may specify more keys to finetune the search.

Returns The first item or a list with all matches.

from_list(*_list*)
Create all collection object items from `_list`.

Parameters **_list** (*list*) – The list with all item dictionaries.

get(*name*)
Return object with name in the collection

Parameters **name** – The name of the object to retrieve from the collection.

Returns The object if it exists. Otherwise None.

remove(*name*, *with_delete=True*, *with_sync=True*, *with_triggers=True*, *recursive=False*, *logger=None*)
Remove an item from collection. This method must be overriden in any subclass.

Parameters

- **name** (*str*) – (item name)
- **with_delete** (*bool*) – (sync and run triggers)
- **with_sync** (*bool*) – (sync to server file system)
- **with_triggers** (*bool*) – (run "on delete" triggers)
- **recursive** (*bool*) – (recursively delete children)
- **logger** – (logger object)

Returns NotImplementedException

rename(*ref*, *newname*, *with_sync=True*, *with_triggers=True*, *logger=None*)
Allows an object "ref" to be given a newname without affecting the rest of the object tree.

Parameters

- **ref** – The reference to the object which should be renamed.
- **newname** – The new name for the object.
- **with_sync** – If a sync should be triggered when the object is renamed.
- **with_triggers** – If triggers should be run when the object is renamed.
- **logger** – The logger to audit the action with.

to_list()
Serialize the collection

Return type list

Returns All elements of the collection as a list.

to_string()
Creates a printable representation of the collection suitable for reading by humans or parsing from scripts. Actually scripts would be better off reading the JSON in the cobbler_collections files directly.

Returns The object as a string representation.

Return type str

cobbler.cobbler_collections.distros module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.cobbler_collections.distros.**Distros**(*collection_mgr*)
Bases: *cobbler.cobbler_collections.collection.Collection*

A distro represents a network bootable matched set of kernels and initrd files.

static collection_type() → str
Returns the string key for the name of the collection (used by serializer etc)

static collection_types() → str
Returns the string key for the plural name of the collection (used by serializer)

factory_produce(*collection_mgr*, *item_dict*)
Return a Distro forged from item_dict

remove(*name*, *with_delete=True*, *with_sync=True*, *with_triggers=True*, *recursive=False*, *logger=None*)
Remove element named 'name' from the collection

cobbler.cobbler_collections.files module

Copyright 2010, Kelsey Hightower Kelsey Hightower <kelsey.hightower@gmail.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.cobbler_collections.files.**Files**(*collection_mgr*)
Bases: *cobbler.cobbler_collections.collection.Collection*

Files provide a container for file resources.

static collection_type() → str
Returns the string key for the name of the collection (used by serializer etc)

static collection_types() → str
Returns the string key for the plural name of the collection (used by serializer)

factory_produce(*collection_mgr*, *item_dict*)
Return a File forged from item_dict

remove(*name*, *with_delete=True*, *with_sync=True*, *with_triggers=True*, *recursive=False*, *logger=None*)
Remove element named 'name' from the collection

cobbler.cobbler_collections.images module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This software may be freely redistributed under the terms of the GNU general public license.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

class cobbler.cobbler_collections.images.**Images**(*collection_mgr*)
Bases: *cobbler.cobbler_collections.collection.Collection*

A image instance represents a ISO or virt image we want to track and repeatedly install. It differs from a answer-file based installation.

static collection_type() → str
Returns the string key for the name of the collection (used by serializer etc)

static collection_types() → str
Returns the string key for the plural name of the collection (used by serializer)

factory_produce(*collection_mgr*, *item_dict*)
Return a Distro forged from item_dict

remove(*name*, *with_delete=True*, *with_sync=True*, *with_triggers=True*, *recursive=True*, *logger=None*)
Remove element named 'name' from the collection

cobbler.cobbler_collections.manager module

Repository of the Cobbler object model

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.cobbler_collections.manager.**CollectionManager**(*api*)
Bases: object

deserialize()
Load all cobbler_collections from disk

Raises ***CX*** – if there is an error in deserialization

distros()
Return the definitive copy of the Distros collection

files()
Return the definitive copy of the Files collection

generate_uid()
Cobbler itself does not use this GUID's though they are provided to allow for easier API linkage with other applications. Cobbler uses unique names in each collection as the object id aka primary key.

Returns A version 4 UUID according to the python implementation of RFC 4122.

get_items(*collection_type*)
Get a full collection of a single type.

Valid Values vor `collection_type` are: "distro", "profile", "repo", "image", "mgmtclass", "package", "file" and "settings".

Parameters **collection_type** – The type of collection to return.

Returns The collection if `collection_type` is valid.

Raises ***CX*** – If the `collection_type` is invalid.

has_loaded = False

images()
Return the definitive copy of the Images collection

mgmtclasses()
Return the definitive copy of the Mgmtclasses collection

packages()
Return the definitive copy of the Packages collection

profiles()
Return the definitive copy of the Profiles collection

repos()
Return the definitive copy of the Repos collection

serialize()
Save all cobbler_collections to disk

serialize_delete(*collection*, *item*)
Delete a collection item from disk

Parameters

- **collection** – collection
- **item** – collection item

serialize_item(*collection*, *item*)
Save a collection item to disk

Parameters

- **collection** – Collection
- **item** – collection item

settings()
Return the definitive copy of the application settings

systems()
Return the definitive copy of the Systems collection

cobbler.cobbler_collections.mgmtclasses module

class cobbler.cobbler_collections.mgmtclasses.**Mgmtclasses**(*collection_mgr*)
Bases: *cobbler.cobbler_collections.collection.Collection*

A mgmtclass provides a container for management resources.

static collection_type() → str
Returns the string key for the name of the collection (used by serializer etc)

static collection_types() → str
Returns the string key for the plural name of the collection (used by serializer)

factory_produce(*config*, *item_dict*)
Return a mgmtclass forged from item_dict

remove(*name*, *with_delete=True*, *with_sync=True*, *with_triggers=True*, *recursive=False*, *logger=None*)
Remove element named 'name' from the collection

cobbler.cobbler_collections.packages module

Copyright 2010, Kelsey Hightower Kelsey Hightower <kelsey.hightower@gmail.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.cobbler_collections.packages.**Packages**(*collection_mgr*)
Bases: *cobbler.cobbler_collections.collection.Collection*

A package provides a container for package resources.

static collection_type() → str
Returns the string key for the name of the collection (used by serializer etc)

static collection_types() → str
Returns the string key for the plural name of the collection (used by serializer)

factory_produce(*collection_mgr*, *item_dict*)
Return a Package forged from item_dict

remove(*name*, *with_delete=True*, *with_sync=True*, *with_triggers=True*, *recursive=False*, *logger=None*)
Remove element named 'name' from the collection

cobbler.cobbler_collections.profiles module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.cobbler_collections.profiles.**Profiles**(*collection_mgr*)
Bases: *cobbler.cobbler_collections.collection.Collection*

A profile represents a distro paired with an automatic OS installation template file.

static collection_type() → str
Returns the string key for the name of the collection (used by serializer etc)

static collection_types() → str
Returns the string key for the plural name of the collection (used by serializer)

factory_produce(*collection_mgr*, *item_dict*)
Return a Distro forged from item_dict

remove(*name*, *with_delete=True*, *with_sync=True*, *with_triggers=True*, *recursive=False*, *logger=None*)
Remove element named 'name' from the collection

cobbler.cobbler_collections.repos module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.cobbler_collections.repos.**Repos**(*collection_mgr*)
Bases: *cobbler.cobbler_collections.collection.Collection*

Repositories in Cobbler are way to create a local mirror of a yum repository. When used in conjunction with a mirrored distro tree (see "cobbler import"), outside bandwidth needs can be reduced and/or eliminated.

static collection_type() → str
Returns the string key for the name of the collection (used by serializer etc)

static collection_types() → str
Returns the string key for the plural name of the collection (used by serializer)

factory_produce(*config*, *item_dict*)
Return a Distro forged from item_dict

remove(*name*, *with_delete=True*, *with_sync=True*, *with_triggers=True*, *recursive=False*, *logger=None*)
Remove element named 'name' from the collection

cobbler.cobbler_collections.systems module

Copyright 2008-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.cobbler_collections.systems.**Systems**(*collection_mgr*)
Bases: *cobbler.cobbler_collections.collection.Collection*

Systems are hostnames/MACs/IP names and the associated profile they belong to.

static collection_type() → str
Returns the string key for the name of the collection (used by serializer etc)

static collection_types() → str
Returns the string key for the plural name of the collection (used by serializer)

factory_produce(*collection_mgr*, *item_dict*)
Return a Distro forged from item_dict

remove(*name*, *with_delete=True*, *with_sync=True*, *with_triggers=True*, *recursive=False*, *logger=None*)
Remove element named 'name' from the collection

Module contents

8.1.3 cobbler.items package

Submodules

cobbler.items.distro module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.items.distro.**Distro**(**args*, ***kwargs*)

Bases: *cobbler.items.item.Item*

A Cobbler distribution object

COLLECTION_TYPE = 'distro'

TYPE_NAME = 'distro'

check_if_valid()

Check if a distro object is valid. If invalid an exception is raised.

get_arch()

Return the architecture of the distribution

Returns Return the current architecture.

get_fields()

Return the list of fields and their properties

get_parent()

Distros don't have parent objects.

get_redhat_management_key()

Get the redhat management key. This is probably only needed if you have spacewalk, uyuni or SUSE Manager running.

Returns The key as a string.

Return type str

make_clone()

Clone a distro object.

Returns The cloned object. Not persisted on the disk or in a database.

set_arch(*arch*)

The field is mainly relevant to PXE provisioning.

Using an alternative distro type allows for dhcpd.conf templating to "do the right thing" with those systems – this also relates to bootloader configuration files which have different syntax for different distro types (because of the bootloaders).

This field is named "arch" because mainly on Linux, we only care about the architecture, though if (in the future) new provisioning types are added, an arch value might be something like "bsd_x86".

Parameters **arch** – The architecture of the operating system distro.

set_boot_loader(*name*)
Set the bootloader for the distro.

Parameters **name** – The name of the bootloader. Must be one of the supported ones.

set_breed(*breed*)
Set the Operating system breed.

Parameters **breed** – The new breed to set.

set_initrd(*initrd*)
Specifies an initrd image. Path search works as in set_kernel. File must be named appropriately.

Parameters **initrd** – The new path to the `initrd`.

set_kernel(*kernel*)
Specifies a kernel. The kernel parameter is a full path, a filename in the configured kernel directory (set in /etc/cobbler.conf) or a directory path that would contain a selectable kernel. Kernel naming conventions are checked, see docs in the utils module for `find_kernel`.

Parameters **kernel** –

Raises ***CX*** – If the kernel was not found

set_os_version(*os_version*)
Set the Operating System Version.

Parameters **os_version** – The new OS Version.

set_redhat_management_key(*management_key*)
Set the redhat management key. This is probably only needed if you have spacewalk, uyuni or SUSE Manager running.

Parameters **management_key** – The redhat management key.

set_remote_boot_initrd(*remote_boot_initrd*)
URL to a remote initrd. If the bootloader supports this feature, it directly tries to retrieve the initrd and boot it. (grub supports tftp and http protocol and server must be an IP).

set_remote_boot_kernel(*remote_boot_kernel*)
URL to a remote kernel. If the bootloader supports this feature, it directly tries to retrieve the kernel and boot it. (grub supports tftp and http protocol and server must be an IP).

set_source_repos(*repos*)
A list of http:// URLs on the Cobbler server that point to yum configuration files that can be used to install core packages. Use by `cobbler import` only.

Parameters **repos** – The list of URLs.

set_supported_boot_loaders(*supported_boot_loaders*)
Some distributions, particularly on powerpc, can only be netbooted using specific bootloaders.

Parameters **supported_boot_loaders** – The bootloaders which are available for being set.

set_tree_build_time(*datestamp*)
Sets the import time of the distro. If not imported, this field is not meaningful.

Parameters **datestamp** – The datestamp to save the builddate. There is an attempt to convert it to a float, so please make sure it is compatible to this.

cobbler.items.file module

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.items.file.**File**(*collection_mgr*, *is_subobject=False*)
Bases: *cobbler.resource.Resource*

A Cobbler file object.

COLLECTION_TYPE = 'file'

TYPE_NAME = 'file'

check_if_valid()
Insure name, path, owner, group, and mode are set. Templates are only required for files, is_dir = False

get_fields()
Return all fields which this class has with its current values.

Returns This is a list with lists.

make_clone()
Clone this file object. Please manually adjust all values yourself to make the cloned object unique.

Returns The cloned instance of this object.

set_is_dir(*is_dir*)
If true, treat file resource as a directory. Templates are ignored.

Parameters **is_dir** – This is the path to check if it is a directory.

cobbler.items.image module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.items.image.**Image**(*collection_mgr*, *is_subobject=False*)
Bases: *cobbler.items.item.Item*

A Cobbler Image. Tracks a virtual or physical image, as opposed to a answer file (autoinst) led installation.

COLLECTION_TYPE = 'image'

TYPE_NAME = 'image'

get_fields()
Return all fields which this class has with its current values.

Returns This is a list with lists.

get_parent()
Images have no parent object.

get_valid_image_types()
Get all valid image types.

Returns A list currently with the values: “direct”, “iso”, “memdisk”, “virt-clone”

make_clone()
Clone this image object. Please manually adjust all value yourself to make the cloned object unique.

Returns The cloned instance of this object.

set_arch(*arch*)
The field is mainly relevant to PXE provisioning. See comments for set_arch in item_distro.py, this works the same.

Parameters **arch** – The new architecture to set.

set_autoinstall(*autoinstall*)
Set the automatic installation file path, this must be a local file.

It may not make sense for images to have automatic installation templates. It really doesn’t. However if the image type is ‘iso’ koan can create a virtual floppy and shove an answer file on it, to script an installation. This may not be a automatic installation template per se, it might be a Windows answer file (SIF) etc.

Parameters **autoinstall** (*str*) – local automatic installation template file path

set_breed(*breed*)
Set the operating system breed with this setter.

Parameters **breed** – The breed of the operating system which is available in the image.

set_file(*filename*)
Stores the image location. This should be accessible on all nodes that need to access it.

Format: can be one of the following: * username:password@hostname:/path/to/the/filename.ext * username@hostname:/path/to/the/filename.ext * hostname:/path/to/the/filename.ext * /path/to/the/filename.ext

Parameters **filename** – The location where the image is stored.

set_image_type(*image_type*)
Indicates what type of image this is. direct = something like “memdisk”, physical only iso = a bootable ISO that pxe’s or can be used for virt installs, virtual only virt-clone = a cloned virtual disk (FIXME: not yet supported), virtual only memdisk = hdd image (physical only)

Parameters **image_type** – One of the four options from above.

set_network_count(*num*)
Setter for the number of networks.

Parameters **num** (*int*) – If None or emtpy will be set to one. Otherwise will be cast to int and then set.

set_os_version(*os_version*)
Set the operating system version with this setter.

Parameters **os_version** – This must be a valid OS-Version.

set_virt_auto_boot(*num*)
Setter for the virtual automatic boot option.

Parameters **num** – May be “0” (disabled) or “1” (enabled)

set_virt_bridge(*vbridge*)
Setter for the virtual bridge which is used.

Parameters **vbridge** – The name of the virtual bridge to use.

set_virt_cpus(*num*)
Setter for the number of virtual cpus.

Parameters **num** – The number of virtual cpu cores.

set_virt_disk_driver(*driver*)
Setter for the virtual disk driver.

Parameters **driver** – The virtual disk driver which will be set.

set_virt_file_size(*num*)
Setter for the virtual file size of the image.

Parameters **num** – Is a non-negative integer (0 means default). Can also be a comma seperated list – for usage with multiple disks

set_virt_path(*path*)
Setter for the virtual path which is used.

Parameters **path** – The path to where the virtual image is stored.

set_virt_ram(*num*)
Setter for the amount of virtual RAM the machine will have.

Parameters **num** – 0 tells Koan to just choose a reasonable default.

set_virt_type(*vtype*)
Setter for the virtual type

Parameters **vtype** – May be one of "qemu", "kvm", "xenpv", "xenfv", "vmware", "vmwarew", "openvz" or "auto".

cobbler.items.item module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This software may be freely redistributed under the terms of the GNU general public license.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

class cobbler.items.item.**Item**(*collection_mgr*, *is_subobject=False*)
Bases: object

An Item is a serializable thing that can appear in a Collection

TYPE_NAME = 'generic'

check_if_valid()
Raise exceptions if the object state is inconsistent

clear(*is_subobject=False*)
Reset this object.

Parameters **is_subobject** – True if this is a subobject, otherwise the default is enough.

converted_cache = {}

dump_vars(*data*, *format=True*)
Dump all variables.

Parameters

- **data** – Unused parameter in this method.
- **format** – Whether to format the output or not.

Returns The raw or formatted data.

find_match(*kwargs*, *no_errors=False*)
Find from a given dict if the item matches the kv-pairs.

Parameters

- **kwargs** – The dict to match for in this item.

- **no_errors** – How strict this matching is.

Returns True if matches or False if the item does not match.

find_match_single_key(*data*, *key*, *value*, *no_errors=False*)
Look if the data matches or not. This is an alternative for `find_match()`.

Parameters

- **data** – The data to search through.
- **key** – The key to look for int the item.
- **value** – The value for the key.
- **no_errors** – How strict this matching is.

Returns Whether the data matches or not.

Return type bool

from_dict(*_dict*)
Modify this object to take on values in `seed_data`.

Parameters **_dict** – This should contain all values which should be updated.

get_children(*sorted=True*)
Get direct children of this object.

Parameters **sorted** – If the list has to be sorted or not.

Returns The list with the children. If no childrens are present an emtpy list is returned.

Return type list

get_conceptual_parent()
The parent may just be a superclass for something like a subprofile. Get the first parent of a different type.

Returns The first item which is conceptually not from the same type.

get_descendants(*sort=False*)
Get objects that depend on this object, i.e. those that would be affected by a cascading delete, etc.

Parameters **sort** – If True the list will be a walk of the tree, e.g., distro -> [profile, sys, sys, profile, sys, sys]

Returns This is a list of all descendants. May be empty if none exist.

get_fields()
Get serializable fields Must be defined in any subclass

classmethod get_from_cache(*ref*)
Get an object from the cache. This is may potentially contain not persisted changes.

Parameters **ref** – The id of the object which is in the cache.

Returns The object if present or an empty dict.

get_parent()
For objects with a tree relationship, what's the parent object?

get_setter_methods()
Get all setter methods which are available in the item.

Returns A dict with all setter methods.

make_clone()
Must be defined in any subclass

classmethod remove_from_cache(*ref*)
Remove an item from the cache.

Parameters **ref** – The object reference id to identify the object.

set_autoinstall_meta(*options*)
A comma delimited list of key value pairs, like 'a=b,c=d,e=f' or a dict. The meta tags are used as input to the templating system to preprocess automatic installation template files.

Parameters **options** – The new options for the automatic installation meta options.

Returns False if this does not succeed.

set_boot_files(*boot_files*)
A comma seperated list of req_name=source_file_path that should be fetchable via tftp.

Parameters **boot_files** – The new value for the boot files used by the item.

Returns False if this does not succeed.

classmethod set_cache(*ref*, *value*)
Add an object to the cache.

Parameters

- **ref** – An object to identify where to add the item to the cache.
- **value** – The object to add to the cache.

set_comment(*comment*)
Setter for the comment of the item.

Parameters **comment** – The new comment. If `None` the comment will be set to an emtpy string.

set_ctime(*ctime*)
Setter for the creation time of the object.

Parameters **ctime** – The new creation time. Especially usefull for replication Cobbler.

set_depth(*depth*)
Setter for depth.

Parameters **depth** – The new value for depth.

set_fetchable_files(*fetchable_files*)
A comma seperated list of virt_name=path_to_template that should be fetchable via tftp or a webserver

Parameters **fetchable_files** – Files which will be made available to external users.

Returns False if this does not succeed.

set_kernel_options(*options*)
Kernel options are a space delimited list, like 'a=b c=d e=f g h i=j' or a dict.

Parameters **options** – The new kernel options as a space delimited list.

set_kernel_options_post(*options*)
Post kernel options are a space delimited list, like 'a=b c=d e=f g h i=j' or a dict.

Parameters **options** – The new kernel options as a space delimited list.

set_mgmt_classes(*mgmt_classes*)
Assigns a list of configuration management classes that can be assigned to any object, such as those used by Puppet's external_nodes feature.

Parameters **mgmt_classes** – The new options for the management classes of an item.

set_mgmt_parameters(*mgmt_parameters*)
A YAML string which can be assigned to any object, this is used by Puppet's external_nodes feature.

Parameters **mgmt_parameters** – The management parameters for an item.

set_mtime(*mtime*)
Setter for the modification time of the object.

Parameters **mtime** – The new modification time.

set_name(*name*)
Set the objects name.

Parameters **name** (*str*) – object name string

Returns True or CX

set_owners(*data*)
The owners field is a comment unless using an authz module that pays attention to it, like authz_ownership, which ships with Cobbler but is off by default.

Parameters **data** – This can be a string or a list which contains all owners.

set_parent(*parent*)
Set the parent object for this object.

Parameters **parent** – The new parent object. This needs to be a descendant in the logical inheritance chain.

set_template_files(*template_files*)
A comma seperated list of source=destination templates that should be generated during a sync.

Parameters **template_files** – The new value for the template files which are used for the item.

Returns False if this does not succeed.

set_uid(*uid*)
Setter for the uid of the item.

Parameters **uid** – The new uid.

sort_key(*sort_fields=[]*)
Convert the item to a dict and sort the data after specific given fields.

Parameters **sort_fields** – The fields to sort the data after.

Returns The sorted data.

to_dict()
This converts everything in this object to a dictionary.

Returns A dictionary with all values present in this object.

to_string()
Convert an item into a string.

Returns The string representation of the object.

cobbler.items.mgmtclass module

class cobbler.items.mgmtclass.**Mgmtclass**(**args*, ***kwargs*)
Bases: *cobbler.items.item.Item*

COLLECTION_TYPE = 'mgmtclass'

TYPE_NAME = 'mgmtclass'

check_if_valid()
Check if this object is in a valid state. This currently checks only if the name is present.

get_fields()
Return all fields which this class has with it's current values.

Returns This is a list with lists.

make_clone()
Clone this file object. Please manually adjust all value yourself to make the cloned object unique.

Returns The cloned instance of this object.

set_class_name(*name*)
Setter for the name of the managementclass.

Parameters **name** (*str*) – The new name of the class. This must not contain "_", "-", ".", ":" or "+".

set_files(*files*)
Setter for the files of the object.

Parameters **files** – A string or list which contains the new files.

set_is_definition(*isdef*)
Setter for property `is_defintion`.

Parameters **isdef** – The new value for the property.

set_packages(*packages*)
Setter for the packages of the managementclass.

Parameters **packages** – A string or list which contains the new packages.

set_params(*params*)
Setter for the params of the managementclass.

Parameters **params** – The new params for the object.

cobbler.items.package module

Copyright 2006-2009, MadHatter Kelsey Hightower <kelsey.hightower@gmail.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.items.package.**Package**(*collection_mgr*, *is_subobject=False*)
Bases: *cobbler.resource.Resource*

COLLECTION_TYPE = 'package'

TYPE_NAME = 'package'

check_if_valid()
Checks if the object is in a valid state. This only checks currently if the name is present.

get_fields()
Return all fields which this class has with its current values.

Returns This is a list with lists.

make_clone()
Clone this package object. Please manually adjust all value yourself to make the cloned object unique.

Returns The cloned instance of this object.

set_installer(*installer*)
Setter for the installer parameter.

Parameters **installer** (*str*) – This parameter will be lowercased regardless of what string you give it.

set_version(*version*)
Setter for the package version.

Parameters **version** (*str*) – They may be anything which is suitable for describing the version of a package. Internally this is a string.

cobbler.items.profile module

class cobbler.items.profile.**Profile**(**args*, ***kwargs*)
Bases: *cobbler.items.item.Item*

A Cobbler profile object.

COLLECTION_TYPE = 'profile'

TYPE_NAME = 'profile'

check_if_valid()
Check if the profile is valid. This checks for an existing name and a distro as a conceptual parent.

get_arch()
Getter of the architecture of the profile or the parent.

Returns The architecture.

get_fields()
Return all fields which this class has with its current values.

Returns This is a list with lists.

get_parent()
Return object next highest up the tree.

get_redhat_management_key()
Getter of the redhat management key of the profile or it's parent.

Returns Returns the redhat_management_key of the profile.

make_clone()
Clone this file object. Please manually adjust all value yourself to make the cloned object unique.

Returns The cloned instance of this object.

set_autoinstall(*autoinstall*)
Set the automatic OS installation template file path, this must be a local file.

Parameters **autoinstall** (*str*) – local automatic installation template path

set_dhcp_tag(*dhcp_tag*)
Setter for the dhcp tag property.

Parameters **dhcp_tag** –

set_distro(*distro_name*)
Sets the distro. This must be the name of an existing Distro object in the Distros collection.

set_enable_gpxe(*enable_gpxe*)
Sets whether or not the profile will use gPXE for booting.

Parameters **enable_gpxe** – New boolean value for enabling gPXE.

set_enable_menu(*enable_menu*)
Sets whether or not the profile will be listed in the default PXE boot menu. This is pretty forgiving for YAML's sake.

Parameters **enable_menu** – New boolean value for enabling the menu.

set_filename(*filename*)

set_name_servers(*data*)
Set the DNS servers.

Parameters **data** – string or list of nameservers

Returns True or throws exception

Raises ***CX*** – If the nameservers are not valid.

set_name_servers_search(*data*)
Set the DNS search paths.

Parameters **data** – string or list of search domains

Returns True or throws exception

Raises ***CX*** – If the search domains are not valid.

set_next_server(*server*)
Setter for the next server value.

Parameters **server** – If this is None or an emtpy string this will be reset to be inherited from the parent object.

set_parent(*parent_name*)
Instead of a `--distro`, set the parent of this object to another profile and use the values from the parent instead of this one where the values for this profile aren't filled in, and blend them together where they are dictionaries. Basically this enables profile inheritance. To use this, the object MUST have been constructed with `is_subobject=True` or the default values for everything will be screwed up and this will likely NOT work. So, API users – make sure you pass `is_subobject=True` into the constructor when using this.

Parameters **parent_name** – The name of the parent object.

set_proxy(*proxy*)
Setter for the proxy.

Parameters **proxy** – The new proxy for the profile.

set_redhat_management_key(*management_key*)
Setter of the redhat management key.

Parameters **management_key** – The value may be reset by setting it to None.

set_repos(*repos*, *bypass_check=False*)
Setter of the repositories for the profile.

Parameters

- **repos** – The new repositories which will be set.
- **bypass_check** – If repository checks should be checked or not.

set_server(*server*)
Setter for the server property.

Parameters **server** – If this is None or an emtpy string this will be reset to be inherited from the parent object.

set_virt_auto_boot(*num*)
Setter for booting a virtual machine automatically.

Parameters **num** – The new value for whether to enable it or not.

set_virt_bridge(*vbridge*)
Setter for the name of the virtual bridge to use.

Parameters **vbridge** – The name of the virtual bridge to use.

set_virt_cpus(*num*)
Setter for the number of virtual CPU cores to assign to the virtual machine.

Parameters **num** – The number of cpu cores.

set_virt_disk_driver(*driver*)
Setter for the virtual disk driver that will be used.

Parameters **driver** – The new driver.

set_virt_file_size(*num*)
Setter for the size of the virtual image size.

Parameters **num** – The new size of the image.

set_virt_path(*path*)
Setter of the path to the place where the image will be stored.

Parameters **path** – The path to where the image will be stored.

set_virt_ram(*num*)
Setter for the virtual RAM used for the VM.

Parameters **num** – The number of RAM to use for the VM.

set_virt_type(*vtype*)
Setter for the virtual machine type.

Parameters **vtype** – May be on out of “qemu”, “kvm”, “xenpv”, “xenfv”, “vmware”, “vmwarew”, “openvz” or “auto”.

cobbler.items.repo module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.items.repo.**Repo**(**args*, ***kwargs*)
Bases: *cobbler.items.item.Item*

A Cobbler repo object.

COLLECTION_TYPE = 'repo'

TYPE_NAME = 'repo'

check_if_valid()
Checks if the object is valid. Currently checks for name and mirror to be present.

get_fields()
Return all fields which this class has with its current values.

Returns This is a list with lists.

get_parent()
Currently the Cobbler object space does not support subobjects of this object as it is conceptually not useful.

make_clone()
Clone this file object. Please manually adjust all value yourself to make the cloned object unique.

Returns The cloned instance of this object.

set_apt_components(*value*)
Setter for the apt command property.

Parameters **value** – The new value for `apt_components`.

set_apt_dists(*value*)
Setter for the apt dists.

Parameters **value** – The new value for `apt_dists`.

Returns `True` if everything went correctly.

set_arch(*arch*)
Override the arch used for reposync

Parameters **arch** – The new arch which will be used.

set_breed(*breed*)
Setter for the operating system breed.

Parameters **breed** – The new breed to set. If this argument evaluates to false then nothing will be done.

set_createrepo_flags(*createrepo_flags*)
Flags passed to createrepo when it is called. Common flags to use would be `-c cache` or `-g comps.xml` to generate group information.

Parameters **createrepo_flags** – The createrepo flags which are passed additionally to the default ones.

set_environment(*options*)
Yum can take options from the environment. This puts them there before each reposync.

Parameters **options** – These are environment variables which are set before each reposync.

set_keep_updated(*keep_updated*)
This allows the user to disable updates to a particular repo for whatever reason.

Parameters **keep_updated** – This may be a bool-like value if the repository shall be keept up to date or not.

set_mirror(*mirror*)
A repo is (initially, as in right now) is something that can be rsynced. reposync/repotrack integration over HTTP might come later.

Parameters **mirror** – The mirror URI.

set_mirror_locally(*value*)
Setter for the local mirror property.

Parameters **value** – The new value for `mirror_locally`.

set_os_version(*os_version*)
Setter for the operating system version.

Parameters **os_version** – The new operating system version. If this argument evaluates to false then nothing will be done.

set_priority(*priority*)
Set the priority of the repository. Only works if host is using priorities plugin for yum.

Parameters **priority** – Must be a value between 1 and 99. 1 is the highest whereas 99 is the default and lowest.

set_proxy(*value*)
Setter for the proxy setting of the repository.

Parameters **value** – The new proxy which will be used for the repository.

Returns `True` if this succeeds.

set_rpm_list(*rpms*)
Rather than mirroring the entire contents of a repository (Fedora Extras, for instance, contains games, and we probably don't want those), make it possible to list the packages one wants out of those repos, so only those packages and deps can be mirrored.

Parameters **rpms** – The rpm to mirror. This may be a string or list.

set_yumopts(*options*)
Kernel options are a space delimited list.

Parameters **options** – Something like 'a=b c=d e=f g h i=j' or a dictionary.

cobbler.items.system module

class cobbler.items.system.**System**(**args*, ***kwargs*)
Bases: *cobbler.items.item.Item*

A Cobbler system object.

COLLECTION_TYPE = 'system'

TYPE_NAME = 'system'

check_if_valid()
Raise exceptions if the object state is inconsistent

delete_interface(*name*)
Used to remove an interface.

from_dict(*seed_data*)
Modify this object to take on values in `seed_data`.

Parameters **_dict** – This should contain all values which should be updated.

get_config_filename(*interface*, *loader=None*)
The configuration file for each system pxe uses is either a form of the MAC address of the hex version of the IP. If none of that is available, just use the given name, though the name given will be unsuitable for PXE configuration (For this, check system.is_management_supported()). This same file is used to store system config information in the Apache tree, so it's still relevant.

Parameters

- **loader** (*str*) – Bootloader type.
- **interface** (*str*) – Name of the interface.

get_fields()
Get serializable fields Must be defined in any subclass

get_ip_address(*interface*)
Get the IP address for the given interface.

get_mac_address(*interface*)
Get the mac address, which may be implicit in the object name or explicit with –mac-address. Use the explicit location first.

get_parent()
Return object next highest up the tree.

get_redhat_management_key()

is_management_supported(*cidr_ok=True*)
Can only add system PXE records if a MAC or IP address is available, else it's a koan only record.

make_clone()
Must be defined in any subclass

modify_interface(*_dict*)
Used by the WUI to modify an interface more-efficiently

rename_interface(*names*)
Used to rename an interface.

set_autoinstall(*autoinstall*)
Set the automatic installation template filepath, this must be a local file.

@param str local automatic installation template file path

set_bonding_opts(*bonding_opts*, *interface*)

set_boot_loader(*name*)

set_bridge_opts(*bridge_opts*, *interface*)

set_cnames(*cnames*, *interface*)

set_connected_mode(*truthiness*, *interface*)

set_dhcp_tag(*dhcp_tag*, *interface*)

set_dns_name(*dns_name*, *interface*)
Set DNS name for interface.

@param: str dns_name (dns name) @param: str interface (interface name) @returns: True or CX

set_enable_gpxe(*enable_gpxe*)
Sets whether or not the system will use gPXE for booting.

set_filename (*filename*)

set_gateway (*gateway*)
Set a gateway IPv4 address.

@param: str gateway (ip address) @returns: True or CX

set_hostname (*hostname*)
Set hostname.

@param: str hostname (hostname for system) @returns: True or CX

set_if_gateway (*gateway*, *interface*)
Set the per-interface gateway.

@param: str gateway (ipv4 address for the gateway) @param: str interface (interface name) @returns: True or CX

set_image (*image_name*)
Set the system to use a certain named image. Works like set_profile but cannot be used at the same time. It's one or the other.

set_interface_master (*interface_master*, *interface*)

set_interface_type (*type*, *interface*)

set_ip_address (*address*, *interface*)
Set IPv4 address on interface.

@param: str address (ip address) @param: str interface (interface name) @returns: True or CX

set_ipv6_address (*address*, *interface*)
Set IPv6 address on interface.

@param: str address (ip address) @param: str interface (interface name) @returns: True or CX

set_ipv6_autoconfiguration (*truthiness*)

set_ipv6_default_device (*interface_name*)

set_ipv6_default_gateway (*address*, *interface*)

set_ipv6_mtu (*mtu*, *interface*)

set_ipv6_prefix (*prefix*, *interface*)
Assign a IPv6 prefix

set_ipv6_secondaries (*addresses*, *interface*)

set_ipv6_static_routes (*routes*, *interface*)

set_mac_address (*address*, *interface*)
Set mac address on interface.

@param: str address (mac address) @param: str interface (interface name) @returns: True or CX

set_management (*truthiness*, *interface*)

set_mtu (*mtu*, *interface*)

set_name_servers (*data*)
Set the DNS servers.

@param: str/list data (string or list of nameservers) @returns: True or CX

set_name_servers_search (*data*)
Set the DNS search paths.

@param: str/list data (string or list of search domains) @returns: True or CX

set_netboot_enabled(*netboot_enabled*)
If true, allows per-system PXE files to be generated on sync (or add). If false, these files are not generated, thus eliminating the potential for an infinite install loop when systems are set to PXE boot first in the boot order. In general, users who are PXE booting first in the boot order won't create system definitions, so this feature primarily comes into play for programmatic users of the API, who want to initially create a system with netboot enabled and then disable it after the system installs, as triggered by some action in automatic installation file's %post section. For this reason, this option is not urfaced in the CLI, output, or documentation (yet).

Use of this option does not affect the ability to use PXE menus. If an admin has machines set up to PXE only after local boot fails, this option isn't even relevant.

set_netmask(*netmask*, *interface*)
Set the netmask for given interface.

@param: str netmask (netmask) @param: str interface (interface name) @returns: True or CX

set_next_server(*server*)

set_power_address(*power_address*)

set_power_id(*power_id*)

set_power_identity_file(*power_identity_file*)

set_power_options(*power_options*)

set_power_pass(*power_pass*)

set_power_type(*power_type*)

set_power_user(*power_user*)

set_profile(*profile_name*)
Set the system to use a certain named profile. The profile must have already been loaded into the Profiles collection.

set_proxy(*proxy*)

set_redhat_management_key(*management_key*)

set_repos_enabled(*repos_enabled*)

set_serial_baud_rate(*baud_rate*)

set_serial_device(*device_number*)

set_server(*server*)
If a system can't reach the boot server at the value configured in settings because it doesn't have the same name on it's subnet this is there for an override.

set_static(*truthiness*, *interface*)

set_static_routes(*routes*, *interface*)

set_status(*status*)

set_virt_auto_boot(*num*)

set_virt_bridge(*bridge*, *interface*)

set_virt_cpus(*num*)

set_virt_disk_driver(*driver*)

set_virt_file_size(*num*)

set_virt_path(*path*)

set_virt_pxe_boot(*num*)

set_virt_ram(*num*)

set_virt_type(*vtype*)

Module contents

8.1.4 cobbler.modules package

Subpackages

cobbler.modules.authentication package

Submodules

cobbler.modules.authentication.configfile module

Authentication module that uses /etc/cobbler/auth.conf Choice of authentication module is in /etc/cobbler/modules.conf

cobbler.modules.authentication.configfile.**authenticate**(*api_handle*, *username*, *password*)

Validate a username/password combo.

Thanks to http://trac.edgewall.org/ticket/845 for supplying the algorithm info.

Parameters

- **api_handle** – Unused in this implementation.
- **username** (*str*) – The username to log in with. Must be contained in /etc/cobbler/users.digest
- **password** (*str*) – The password to log in with. Must be contained hashed in /etc/cobbler/users.digest

Returns A boolean which contains the information if the username/password combination is correct.

Return type bool

cobbler.modules.authentication.configfile.**hashfun**(*text*)

Converts a str object to a hash which was configured in modules.conf of the Cobbler settings.

Parameters **text** (*str*) – The text to hash.

Returns The hash of the text. This should output the same hash when entered the same text.

cobbler.modules.authentication.configfile.**register**()

The mandatory Cobbler module registration hook.

cobbler.modules.authentication.denyall module

Authentication module that denies everything. Used to disable the WebUI by default.

`cobbler.modules.authentication.denyall.`**`authenticate`**(*api_handle*, *username*, *password*)

Validate a username/password combo, returning True/False

Thanks to http://trac.edgewall.org/ticket/845 for supplying the algorithm info.

`cobbler.modules.authentication.denyall.`**`register`**()

The mandatory Cobbler module registration hook.

cobbler.modules.authentication.ldap module

Authentication module that uses ldap Settings in /etc/cobbler/authn_ldap.conf Choice of authentication module is in /etc/cobbler/modules.conf

`cobbler.modules.authentication.ldap.`**`authenticate`**(*api_handle*, *username*, *password*)

Validate an LDAP bind, returning whether the authentication was successful or not.

Parameters

- **`api_handle`** – The api instance to resolve settings.
- **`username`** – The username to authenticate.
- **`password`** – The password to authenticate.

Returns True if the ldap server authentication was a success, otherwise false.

`cobbler.modules.authentication.ldap.`**`register`**()

The mandatory Cobbler module registration hook.

Returns Always “authn”

Return type str

cobbler.modules.authentication.pam module

Authentication module that uses /etc/cobbler/auth.conf Choice of authentication module is in /etc/cobbler/modules.conf

PAM python code based on the pam_python code created by Chris AtLee: http://atlee.ca/software/pam/

PAM module for python

Provides an authenticate function that will allow the caller to authenticate a user against the Pluggable Authentication Modules (PAM) on the system.

Implemented using ctypes, so no compilation is necessary.

class cobbler.modules.authentication.pam.**PamConv**
Bases: _ctypes.Structure

wrapper class for pam_conv structure

appdata_ptr
Structure/Union member

conv
Structure/Union member

class cobbler.modules.authentication.pam.**PamHandle**
Bases: _ctypes.Structure

wrapper class for pam_handle_t

handle
Structure/Union member

class cobbler.modules.authentication.pam.**PamMessage**
Bases: _ctypes.Structure

wrapper class for pam_message structure

msg
Structure/Union member

msg_style
Structure/Union member

class cobbler.modules.authentication.pam.**PamResponse**
Bases: _ctypes.Structure

wrapper class for pam_response structure

resp
Structure/Union member

resp_retcode
Structure/Union member

cobbler.modules.authentication.pam.**authenticate**(*api_handle*, *username*, *password*)
Returns True if the given username and password authenticate for the given service. Returns False otherwise

cobbler.modules.authentication.pam.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.authentication.passthru module

Authentication module that defers to Apache and trusts what Apache trusts.

Copyright 2008-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This software may be freely redistributed under the terms of the GNU general public license.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

cobbler.modules.authentication.passthru.**authenticate**(*api_handle*, *username*, *password*)
Validate a username/password combo. Uses cobbler_auth_helper

Parameters

- **api_handle** – This parameter is not used currently.
- **username** – This parameter is not used currently.
- **password** – This should be the internal Cobbler secret.

Returns True if the password is the secret, otherwise false.

Return type bool

cobbler.modules.authentication.passthru.**register**()
The mandatory Cobbler module registration hook.

Returns Always "authn"

Return type str

cobbler.modules.authentication.spacewalk module

Authentication module that uses Spacewalk's auth system. Any org_admin or kickstart_admin can get in.

Copyright 2007-2008, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.modules.authentication.spacewalk.**authenticate**(*api_handle*, *username*, *password*)
Validate a username/password combo, returning True/False

This will pass the username and password back to Spacewalk to see if this authentication request is valid.

See also: https://github.com/uyuni-project/uyuni/blob/bbbbbf537a1928c1922015c70322034a89b1cb9a/java/code/src/com/redhat/rhn/frontend/xmlrpc/auth/AuthHandler.java#L133

Parameters

- **api_handle** – The api instance to retrieve settings of.
- **username** – The username to authenticate agains spacewalk/uyuni/SUSE Manager
- **password** – The password to authenticate agains spacewalk/uyuni/SUSE Manager

Returns True if it succeeded, False otherwise.

Return type bool

cobbler.modules.authentication.spacewalk.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.authentication.testing module

Authentication module that denies everything. Unsafe demo. Allows anyone in with testing/testing.

Copyright 2007-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.modules.authentication.testing.**authenticate**(*api_handle*, *username*, *password*)
Validate a username/password combo, returning True/False

Thanks to http://trac.edgewall.org/ticket/845 for supplying the algorithm info.

Parameters

- **api_handle** – This parameter is not used currently.
- **username** – The username which should be checked.
- **password** – The password which should be checked.

Returns True if username is “testing” and password is “testing”. Otherwise False.

Return type bool

cobbler.modules.authentication.testing.**register**()
The mandatory Cobbler module registration hook.

Returns Always “authn”

Return type str

Module contents

cobbler.modules.authorization package

Submodules

cobbler.modules.authorization.allowall module

Authorization module that allows everything, which is the default for new Cobbler installs.

Copyright 2007-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

`cobbler.modules.authorization.allowall.`**`authorize`**(*api_handle*, *user*, *resource*, *arg1=None*, *arg2=None*)

Validate a user against a resource. NOTE: acls are not enforced as there is no group support in this module

Parameters

- **`api_handle`** – This parameter is not used currently.
- **`user`** – This parameter is not used currently.
- **`resource`** – This parameter is not used currently.
- **`arg1`** – This parameter is not used currently.
- **`arg2`** – This parameter is not used currently.

Returns Always True

Return type bool

`cobbler.modules.authorization.allowall.`**`register`**()

The mandatory Cobbler module registration hook.

Returns Always "authz"

Return type str

cobbler.modules.authorization.configfile module

Authorization module that allow users listed in /etc/cobbler/users.conf to be permitted to access resources. For instance, when using authz_ldap, you want to use authn_configfile, not authz_allowall, which will most likely NOT do what you want.

This software may be freely redistributed under the terms of the GNU general public license.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

`cobbler.modules.authorization.configfile.`**`authorize`**(*api_handle*, *user*, *resource*, *arg1=None*, *arg2=None*)

Validate a user against a resource. All users in the file are permitted by this module.

Parameters

- **`api_handle`** – This parameter is not used currently.
- **`user`** – The user to authorize.
- **`resource`** – This parameter is not used currently.
- **`arg1`** – This parameter is not used currently.
- **`arg2`** – This parameter is not used currently.

Returns "0" if no authorized, "1" if authorized.

`cobbler.modules.authorization.configfile.`**`register`**()

The mandatory Cobbler module registration hook.

Returns Always “authz”.

Return type str

cobbler.modules.authorization.ownership module

Authorization module that allow users listed in /etc/cobbler/users.conf to be permitted to access resources, with the further restriction that Cobbler objects can be edited to only allow certain users/groups to access those specific objects.

Copyright 2008-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

`cobbler.modules.authorization.ownership.`**`authorize`**(*api_handle*, *user*, *resource*, *arg1=None*, *arg2=None*)

Validate a user against a resource. All users in the file are permitted by this module.

Parameters

- **`api_handle`** – The api to resolve required information.
- **`user`** – The user to authorize to the resource.
- **`resource`** – The resource the user is asking for access. This is something abstract like a remove operation.
- **`arg1`** – This is normally the name of the specific object in question.
- **`arg2`** – This parameter is pointless currently. Reserved for future code.

Returns `True` or `1` if okay, otherwise `False`.

`cobbler.modules.authorization.ownership.`**`register`**()

The mandatory Cobbler module registration hook.

Returns Always “authz”

Return type str

Module contents

cobbler.modules.installation package

Submodules

cobbler.modules.installation.post_log module

(C) 2008-2009, Red Hat Inc. Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.modules.installation.post_log.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.installation.post_log.**run**(*api*, *args*, *logger*)

Parameters

- **api** – This parameter is unused currently.
- **args** – An array of three elements. Type (system/profile), name and ip. If no ip is present use a ?.
- **logger** – This parameter is unused currently.

Returns Always 0

cobbler.modules.installation.post_power module

class cobbler.modules.installation.post_power.**reboot**(*api*, *target*)
Bases: threading.Thread

run()
Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

cobbler.modules.installation.post_power.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.installation.post_power.**run**(*api*, *args*, *logger*)
Obligatory trigger hook.

Parameters

- **api** – The api to resolve information with.
- **args** – This is an array containing two objects. 0: String with the content "target" or "profile". 1: The name of target or profile
- **logger** – Unused parameter for this hook.

Returns 0 on success.

cobbler.modules.installation.post_puppet module

This module signs newly installed client puppet certificates if the puppet master server is running on the same machine as the Cobbler server.

Based on: http://www.ithiriel.com/content/2010/03/29/writing-install-triggers-cobbler

cobbler.modules.installation.post_puppet.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.installation.post_puppet.**run**(*api*, *args*, *logger*)
The obligatory Cobbler modules hook.

Parameters

- **api** – The api to resolve all information with.
- **args** – This is an array with two items. The first may be `system` or `profile` and the second is the name of this system or profile.
- **logger** – The logger to audit all actions with.

Returns `0` or nothing.

cobbler.modules.installation.post_report module

`cobbler.modules.installation.post_report.`**register**`()`
The mandatory Cobbler module registration hook.

Returns Always `/var/lib/cobbler/triggers/install/post/*`.

`cobbler.modules.installation.post_report.`**run**`(`*api*, *args*, *logger*`)`
This is the mandatory Cobbler module run trigger hook.

Parameters

- **api** – The api to resolve information with.
- **args** – This is an array with three elements. 0: "target" or "profile" 1: name of target or profile 2: ip or "?"
- **logger** – In this module not used.

Returns `0` or `1`.

cobbler.modules.installation.pre_clear_anamon_logs module

(C) 2008-2009, Red Hat Inc. James Laska <jlaska@redhat.com> Bill Peck <bpeck@redhat.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

`cobbler.modules.installation.pre_clear_anamon_logs.`**register**`()`
This pure python trigger acts as if it were a legacy shell-trigger, but is much faster. The return of this method indicates the trigger type.

Returns Always: "/var/lib/cobbler/triggers/install/pre/*"

Return type str

`cobbler.modules.installation.pre_clear_anamon_logs.`**run**`(`*api*, *args*, *logger*`)`

The list of args should have one element:

- 1: the name of the system or profile

Parameters

- **api** – The api to resolve metadata with.
- **args** – This should be a list as described above.
- **logger** – This parameter is unused currently.

Returns "0" on success.

cobbler.modules.installation.pre_log module

cobbler.modules.installation.pre_log.**register**()
This pure python trigger acts as if it were a legacy shell-trigger, but is much faster. The return of this method indicates the trigger type.

Returns Always: "/var/lib/cobbler/triggers/install/pre/*"

Return type str

cobbler.modules.installation.pre_log.**run**(*api*, *args*, *logger*)
The method runs the trigger, meaning this logs that an installation has started.

The list of args should have three elements:

- 0: system or profile
- 1: the name of the system or profile
- 2: the ip or a "?"

Parameters

- **api** – This parameter is currently unused.
- **args** (*list*) – Already described above.
- **logger** – This parameter is currently unused.

Returns A "0" on success.

cobbler.modules.installation.pre_puppet module

This module removes puppet certs from the puppet master prior to reinstalling a machine if the puppet master is running on the Cobbler server.

Based on: http://www.ithiriel.com/content/2010/03/29/writing-install-triggers-cobbler

cobbler.modules.installation.pre_puppet.**register**()
This pure python trigger acts as if it were a legacy shell-trigger, but is much faster. The return of this method indicates the trigger type.

Returns Always: "/var/lib/cobbler/triggers/install/pre/*"

Return type str

cobbler.modules.installation.pre_puppet.**run**(*api*, *args*, *logger*)
This method runs the trigger, meaning in this case that old puppet certs are automatically removed via puppetca.

The list of args should have two elements:

- 0: system or profile
- 1: the name of the system or profile

Parameters

- **api** – The api to resolve external information with.
- **args** – Already described above.
- **logger** – The logger to audit the action with.

Returns "0" on success. If unsuccessful this raises an exception.

Module contents

cobbler.modules.managers package

Submodules

cobbler.modules.managers.bind module

This is some of the code behind 'cobbler sync'.

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail> John Eckersberg <jeckersb@redhat.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.modules.managers.bind.**BindManager**(*collection_mgr*, *logger*)

Bases: object

regen_hosts()

Not used.

what()

Identifies what this class is managing.

Returns Always will return bind.

write_dns_files()

BIND files are written when manage_dns is set in /var/lib/cobbler/settings.

cobbler.modules.managers.bind.**get_manager**(*collection_mgr*, *logger*)

This returns the object to manage a BIND server located locally on the Cobbler server.

Parameters

- **collection_mgr** – The collection manager to resolve all information with.
- **logger** – The logger to audit all actions with.

Returns The BindManger object to manage bind with.

cobbler.modules.managers.bind.**register**()

The mandatory Cobbler module registration hook.

cobbler.modules.managers.dnsmasq module

This is some of the code behind 'cobbler sync'.

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail> John Eckersberg <jeckersb@redhat.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.modules.managers.dnsmasq.**DnsmasqManager**(*collection_mgr*, *logger*, *dhcp=None*)

Bases: object

Handles conversion of internal state to the tftpboot tree layout.

regen_ethers()
This function regenerates the ethers file. To get more information please read man ethers, the format is also in there described.

regen_hosts()
This rewrites the hosts file and thus also rewrites the dns config.

remove_dhcp_lease(*port*, *host*)
Not used

Parameters

- **port** – Unused in this module implementation.
- **host** – Unused in this module implementation.

sync_dhcp()
This restarts the dhcp server and thus applied the newly written config files.

what()
This identifies the module.

Returns Will always return dnsmasq.

write_dhcp_file()
DHCP files are written when manage_dhcp is set in /etc/cobbler/settings.

write_dhcp_lease(*port*, *host*, *ip*, *mac*)
Not used

Parameters

- **port** – Unused in this module implementation.
- **host** – Unused in this module implementation.
- **ip** – Unused in this module implementation.
- **mac** – Unused in this module implementation.

write_dns_files()
Not used

cobbler.modules.managers.dnsmasq.**get_manager**(*collection_mgr*, *logger*)
Creates a manager object to manage a dnsmasq server.

Parameters

- **collection_mgr** – The collection manager to resolve all information with.
- **logger** – The logger to audit all actions with.

Returns The object generated from the class.

cobbler.modules.managers.dnsmasq.**register**()
The mandatory Cobbler modules registration hook.

Returns Always "manage".

cobbler.modules.managers.genders module

cobbler.modules.managers.genders.**register**()
We should run anytime something inside of Cobbler changes.

Returns Always /var/lib/cobbler/triggers/change/*

cobbler.modules.managers.genders.**run**(*api*, *args*, *logger*)
Mandatory Cobbler trigger hook.

Parameters

- **api** – The api to resolve information with.
- **args** – For this implementation unused.
- **logger** – The logger to audit all actions with.

Returns 0 or 1, depending on the outcome of the operation.

cobbler.modules.managers.genders.**write_genders_file**(*config*, *profiles_genders*, *distros_genders*, *mgmtcls_genders*)
Genders file is over-written when manage_genders is set in /var/lib/cobbler/settings.

Parameters

- **config** – The config file to template with the data.
- **profiles_genders** – The profiles which should be included.
- **distros_genders** – The distros which should be included.
- **mgmtcls_genders** – The management classes which should be included.

cobbler.modules.managers.import_signatures module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail> John Eckersberg <jeckersb@redhat.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.modules.managers.import_signatures.**ImportSignatureManager**(*collection_mgr*, *logger*)
Bases: object

add_entry(*dirname*, *kernel*, *initrd*)
When we find a directory with a valid kernel/initrd in it, create the distribution objects as appropriate and save them. This includes creating xen and rescue distros/profiles if possible.

Parameters

- **dirname** – Unkown what this currently does.
- **kernel** – Unkown what this currently does.
- **initrd** – Unkown what this currently does.

Returns Unkown what this currently does.

apt_repo_adder(*distro*)
Automatically import apt repositories when importing signatures.

Parameters **distro** – The distribution to scan for apt repositories.

arch_walker(*foo*, *dirname*, *fnames*)
Function for recursively searching through a directory for a kernel file matching a given architecture, called by `learn_arch_from_tree()`

Parameters

- **foo** (*dict*) – Into this dict there will be put additional meta information.
- **dirname** – The directory name where the kernel can be found.
- **fnames** – This should be a list like object which will be looped over.

configure_tree_location(*distro*)
Once a distribution is identified, find the part of the distribution that has the URL in it that we want to use for automating the Linux distribution installation, and create a autoinstall_meta variable $tree that contains this.

Parameters **distro** – The distribution object for that the tree should be configured.

distro_adder(*distros_added*, *dirname*, *fnames*)
This is an import_walker routine that finds distributions in the directory to be scanned and then creates them.

Parameters

- **distros_added** – Unkown what this currently does.
- **dirname** – Unkown what this currently does.
- **fnames** – Unkown what this currently does.

get_file_lines(*filename*)
Get lines from a file, which may or may not be compressed. If compressed then it will be uncompressed using `gzip` as the algorithm.

Parameters **filename** – The name of the file to be read.

Returns An array with all the lines.

get_proposed_name(*dirname*, *kernel=None*)
Given a directory name where we have a kernel/initrd pair, try to autoname the distribution (and profile) object based on the contents of that path.

Parameters

- **dirname** – The directory where the distribution is living in.
- **kernel** – The kernel of that distro.

Returns The name which is recommended.

Return type str

get_repo_mirror_from_apt()
This tries to determine the apt mirror/archive to use (when processing repos) if the host machine is Debian or Ubuntu.

Returns False if the try fails or otherwise the mirrors.

get_valid_arches()
Get all valid architectures from the signature file.

Returns An empty list or all valid architectures.

get_valid_repo_breeds()
Get all valid repository architectures from the signatures file.

Returns An empty list or all valid architectures.

learn_arch_from_tree()
If a distribution is imported from DVD, there is a good chance the path doesn't contain the arch and we should add it back in so that it's part of the meaningful name ... so this code helps figure out the arch name. This is important for producing predictable distro names (and profile names) from differing import sources.

Returns The guessed architecture from a distribution dvd.

Return type list

repo_finder(*distros_added*)
This routine looks through all distributions and tries to find any applicable repositories in those distributions for post-install usage.

Parameters **distros_added** – This is an iteratable set of distributions.

rhn_repo_adder(*distro*)
Not currently used.

Parameters **distro** – Not used currently.

rsync_repo_adder(*distro*)
Not currently used.

Parameters **distro** – Not used currently.

run(*path*, *name*, *network_root=None*, *autoinstall_file=None*, *arch=None*, *breed=None*, *os_version=None*)
This is the main entry point in a manager. It is a required function for import modules.

Parameters

- **path** – the directory we are scanning for files
- **name** – the base name of the distro
- **network_root** – the remote path (nfs/http/ftp) for the distro files
- **autoinstall_file** – user-specified response file, which will override the default
- **arch** – user-specified architecture
- **breed** – user-specified breed
- **os_version** – user-specified OS version

scan_signatures()
Loop through the signatures, looking for a match for both the signature directory and the version file.

set_install_tree(*distro*, *url*)
Simple helper function to set the tree automated installation metavariable.

Parameters

- **distro** – The distribution object for which the install tree should be set.
- **url** – The url for the tree.

what()
Identifies what service this manages.

Returns Always will return `import/signatures`.

yum_process_comps_file(*comps_path*, *distro*)
When importing Fedora/EL certain parts of the install tree can also be used as yum repos containing packages that might not yet be available via updates in yum. This code identifies those areas. Existing repodata will be used as-is, but repodate is created for earlier, non-yum based, installers.

Parameters

- **comps_path** – Not know what this is exactly for.
- **distro** – The distributions to check.

yum_repo_adder(*distro*)
For yum, we recursively scan the rootdir for repos to add

Parameters **distro** – The distribution object to scan and possibly add.

yum_repo_scanner(*distro*, *dirname*, *fnames*)
This is an import_walker routine that looks for potential yum repositories to be added to the configuration for post-install usage.

Parameters

- **distro** – The distribution object to check for.
- **dirname** – The folder with repositories to check.
- **fnames** – Unkown what this does exactly.

cobbler.modules.managers.import_signatures.**get_import_manager**(*config*, *logger*)
Get an instance of the import manager which enables you to import various things.

Parameters

- **config** – The configuration for the import manager.
- **logger** – The logger to audit all actions with.

Returns The object to import data with.

cobbler.modules.managers.import_signatures.**import_walker**(*top*, *func*, *arg*)
Directory tree walk with callback function.

For each directory in the directory tree rooted at top (including top itself, but excluding '.' and '..'), call `func(arg, dirname, fnames)`. dirname is the name of the directory, and fnames a list of the names of the files and subdirectories in dirname (excluding '.' and '..'). `func` may modify the `fnames` list in-place (e.g. via `del` or `slice` assignment), and walk will only recurse into the subdirectories whose names remain in `fnames`; this can be used to implement a filter, or to impose a specific order of visiting. No semantics are defined for, or required of, `arg`, beyond that arg is always passed to `func`. It can be used, e.g., to pass a filename pattern, or a mutable object designed to accumulate statistics.

Parameters

- **top** – The most top directory for which func should be run.
- **func** – A function which is called as described in the above description.
- **arg** – Passing `None` for this is common.

cobbler.modules.managers.import_signatures.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.managers.in_tftpd module

This is some of the code behind 'cobbler sync'.

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.modules.managers.in_tftpd.**InTftpdManager**(*collection_mgr*, *logger*)
Bases: object

add_single_distro(*distro*)

add_single_system(*system*)
Write out new pxelinux.cfg files to /tftpboot

Parameters **system** – The system to be added.

regen_hosts()
Not used

sync(*verbose=True*)
Write out all files to /tftpdboot

Parameters **verbose** – Whether the tftp server should log this verbose or not.

update_netboot(*name*)
Write out new pxelinux.cfg files to /tftpboot

Parameters **name** – The name of the system to update.

what()
Static method to identify the manager.

Returns Always "in_tftpd".

write_boot_files()
Copy files in profile["boot_files"] into /tftpboot. Used for vmware currently.

Returns 0 on success.

write_boot_files_distro(*distro*)

write_dns_files()
Not used

cobbler.modules.managers.in_tftpd.**get_manager**(*collection_mgr*, *logger*)
Creates a manager object to manage an in_tftp server.

Parameters

- **collection_mgr** – The collection manager which holds all information in the current Cobbler instance.
- **logger** – The logger to audit all actions with.

Returns The object to manage the server with.

cobbler.modules.managers.in_tftpd.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.managers.isc module

This is some of the code behind 'cobbler sync'.

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail> John Eckersberg <jeckersb@redhat.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.modules.managers.isc.**IscManager**(*collection_mgr*, *logger*)
Bases: object

regen_ethers()
ISC/BIND doesn't use this. It is there for compability reasons with other managers.

sync_dhcp()
This syncs the dhcp server with it's new config files. Basically this restarts the service to apply the changes.

what()
Static method to identify the manager.

Returns Always "isc".

write_dhcp_file()
DHCP files are written when manage_dhcp is set in /etc/cobbler/settings.

cobbler.modules.managers.isc.**get_manager**(*collection_mgr*, *logger*)
Creates a manager object to manage an isc dhcp server.

Parameters

- **collection_mgr** – The collection manager which holds all information in the current Cobbler instance.
- **logger** – The logger to audit all actions with.

Returns The object to manage the server with.

cobbler.modules.managers.isc.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.managers.ndjbdns module

This is some of the code behind 'cobbler sync'.

Copyright 2014, Mittwald CM Service GmbH & Co. KG Martin Helmich <m.helmich@mittwald.de> Daniel Krämer <d.kraemer@mittwald.de>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.modules.managers.ndjbdns.**NDjbDnsManager**(*config*, *logger*)
Bases: object

regen_hosts()
Empty stub method to have compability with other dns managers who need this.

what()
Static method to identify the manager.

Returns Always "ndjbdns".

write_dns_files()
This writes the new dns configuration file to the disc.

cobbler.modules.managers.ndjbdns.**get_manager**(*config*, *logger*)
Get the DNS Manger object.

Parameters

- **config** – Unused parameter.
- **logger** – The logger to audit the actions with.

Returns The manager object.

cobbler.modules.managers.ndjbdns.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.managers.tftpd_py module

This is some of the code behind 'cobbler sync'.

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.modules.managers.tftpd_py.**TftpdPyManager**(*collection_mgr*, *logger*)
Bases: object

add_single_distro(*distro*)
This function is just a stub for compliance with the other tftp server.

Parameters **distro** – This parameter is unused.

add_single_system(*name*)
Write out files to /tftpdboot. Unused for the python server

regen_hosts()
This function is just a stub for compliance with the other tftp server.

sync(*verbose=True*)
Write out files to /tftpdboot. Mostly unused for the python server

Parameters **verbose** – This parameter is unused.

update_netboot(*name*)
Write out files to /tftpdboot. Unused for the python server

what()

write_boot_files()
Copy files in profile["boot_files"] into /tftpboot. Used for vmware currently.

write_boot_files_distro(*distro*)
Copy files in profile["boot_files"] into /tftpboot. Used for vmware currently.

write_dns_files()
This function is just a stub for compliance with the other tftp server.

cobbler.modules.managers.tftpd_py.**get_manager**(*collection_mgr*, *logger*)
Get the manager object for the tftp server.

Parameters

- **collection_mgr** – The instance who holds all information about Cobbler.

- **logger** – The logger to audit the actions.

Returns The tftp manager instance.

cobbler.modules.managers.tftpd_py.**register**()
The mandatory Cobbler module registration hook.

Module contents

cobbler.modules.serializers package

Submodules

cobbler.modules.serializers.file module

Cobbler's file-based object serializer. As of 9/2014, this is Cobbler's default serializer and the most stable one. It uses multiple JSON files in /var/lib/cobbler/collections/distros, profiles, etc

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.modules.serializers.file.**deserialize**(*collection*, *topological=True*)
Load a collection from file system.

Parameters

- **collection** – The collection type the deserialize
- **topological** (*bool*) – If the dict/list should be sorted or not.

cobbler.modules.serializers.file.**deserialize_raw**(*collection_types*)
Loads a collection from the disk.

Parameters **collection_types** – The type of collection to load.

Returns The loaded dictionary.

cobbler.modules.serializers.file.**filter_upgrade_duplicates**(*file_list*)
In a set of files, some ending with .json, some not, return the list of files with the .json ones taking priority over the ones that are not.

Parameters **file_list** – The list of files to remove duplicates from.

Returns The filtered list of files. Normally this should only return `.json`-Files.

cobbler.modules.serializers.file.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.serializers.file.**serialize**(*collection*)
Save a collection to file system

Parameters **collection** – collection

cobbler.modules.serializers.file.**serialize_delete**(*collection*, *item*)
Delete a collection item from file system.

Parameters

- **collection** – collection
- **item** – collection item

cobbler.modules.serializers.file.**serialize_item**(*collection*, *item*)
Save a collection item to file system

Parameters

- **collection** – collection
- **item** – collection item

cobbler.modules.serializers.file.**what**()
Module identification function

cobbler.modules.serializers.mongodb module

Cobbler's Mongo database based object serializer. Experimental version.

cobbler.modules.serializers.mongodb.**deserialize**(*collection*, *topological=True*)
Load a collection from the database.

Parameters

- **collection** – The collection to deserialize.
- **topological** (*bool*) – This sorts the returned dict.

cobbler.modules.serializers.mongodb.**deserialize_raw**(*collection_type*)
Get a collection from mongodb and parse it into an object.

Parameters **collection_type** – The collection type to fetch.

Returns The first element of the collection requested.

cobbler.modules.serializers.mongodb.**register**()
The mandatory Cobbler module registration hook.

cobbler.modules.serializers.mongodb.**serialize**(*collection*)
Save a collection to database

Parameters **collection** – collection

cobbler.modules.serializers.mongodb.**serialize_delete**(*collection*, *item*)
Delete a collection item from database.

Parameters

- **collection** – collection
- **item** – collection item

cobbler.modules.serializers.mongodb.**serialize_item**(*collection*, *item*)
Save a collection item to database.

Parameters

- **collection** – collection
- **item** – collection item

cobbler.modules.serializers.mongodb.**what**()
Module identification function

Module contents

Submodules

cobbler.modules.nsupdate_add_system_post module

cobbler.modules.nsupdate_add_system_post.**nslog**(*msg*)
Log a message to the logger.

Parameters **msg** – The message to log.

cobbler.modules.nsupdate_add_system_post.**register**()
This method is the obligatory Cobbler registration hook.

Returns The trigger name or an empty string.

Return type str

cobbler.modules.nsupdate_add_system_post.**run**(*api*, *args*, *logger*)
This method executes the trigger, meaning in this case that it updates the dns configuration.

Parameters

- **api** – The api to read metadata from.
- **args** – Metadata to log.
- **logger** – The logger to audit the action with.

Returns "0" on success or a skipped task. If the task failed or problems occurred then an exception is raised.

cobbler.modules.nsupdate_delete_system_pre module

cobbler.modules.nsupdate_delete_system_pre.**nslog**(*msg*)
Log a message to the logger.

Parameters **msg** – The message to log.

cobbler.modules.nsupdate_delete_system_pre.**register**()
This method is the obligatory Cobbler registration hook.

Returns The trigger name or an empty string.

Return type str

cobbler.modules.nsupdate_delete_system_pre.**run**(*api*, *args*, *logger*)
This method executes the trigger, meaning in this case that it updates the dns configuration.

Parameters

- **api** – The api to read metadata from.
- **args** – Metadata to log.
- **logger** – The logger to audit the action with.

Returns "0" on success or a skipped task. If the task failed or problems occurred then an exception is raised.

cobbler.modules.scm_track module

(C) 2009, Red Hat Inc. Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

`cobbler.modules.scm_track.`**`register`**`()`
This pure python trigger acts as if it were a legacy shell-trigger, but is much faster. The return of this method indicates the trigger type :return: Always: `/var/lib/cobbler/triggers/change/*` :rtype: str

`cobbler.modules.scm_track.`**`run`**`(`*api*, *args*, *logger*`)`
Runs the trigger, meaning in this case track any changed which happen to a config or data file.

Parameters

- **`api`** – The api instance of the Cobbler server. Used to look up if scm_track_enabled is true.
- **`args`** – The parameter is currently unused for this trigger.
- **`logger`** – The logger to audit the action with.

Returns 0 on success, otherwise an exception is risen.

cobbler.modules.sync_post_restart_services module

`cobbler.modules.sync_post_restart_services.`**`register`**`()`
This pure python trigger acts as if it were a legacy shell-trigger, but is much faster. The return of this method indicates the trigger type

Returns Always `/var/lib/cobbler/triggers/sync/post/*`

Return type str

`cobbler.modules.sync_post_restart_services.`**`run`**`(`*api*, *args*, *logger*`)`
Run the trigger via this method, meaning in this case that depending on the settings dns and/or dhcp services are restarted.

Parameters

- **`api`** – The api to resolve settings.
- **`args`** – This parameter is not used currently.
- **`logger`** – The logger to audit the action with.

Returns The return code of the service restarts.

Return type int

Module contents

8.1.5 cobbler.web package

Subpackages

cobbler.web.templatetags package

Submodules

cobbler.web.templatetags.site module

class cobbler.web.templatetags.site.**And**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**BaseCalc**(*var1*, *var2=None*, *negate=False*)
Bases: object

calculate(*var1*, *var2*)

resolve(*context*)

resolve_vars(*context*)

class cobbler.web.templatetags.site.**Equals**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**Greater**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**GreaterOrEqual**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**IfParser**(*tokens*)
Bases: object

at_end()

create_var(*value*)

error_class
alias of builtins.ValueError

get_token()

get_var()

parse()

tokens

class cobbler.web.templatetags.site.**In**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**Or**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**SmartIfNode**(*var*, *nodelist_true*, *nodelist_false=None*)
Bases: django.template.base.Node

get_nodes_by_type(*nodetype*)
Return a list of all nodes (within this node and its nodelist) of the given type

render(*context*)
Return the node rendered as a string.

class cobbler.web.templatetags.site.**SmartIfTests**(*methodName='runTest'*)
Bases: unittest.case.TestCase

assertCalc(*calc*, *context=None*)
Test a calculation is True, also checking the inverse "negate" case.

assertCalcFalse(*calc*, *context=None*)
Test a calculation is False, also checking the inverse "negate" case.

setUp()
Hook method for setting up the test fixture before exercising it.

test_and()

test_boolean()

test_equals()

test_greater()

test_greater_or_equal()

test_in()

test_or()

test_parse_bits()

class cobbler.web.templatetags.site.**TemplateIfParser**(*parser*, **args*, ***kwargs*)
Bases: *cobbler.web.templatetags.site.IfParser*

create_var(*value*)

error_class
alias of django.template.exceptions.TemplateSyntaxError

class cobbler.web.templatetags.site.**TestVar**(*value*)
Bases: object

A basic self-resolvable object similar to a Django template variable. Used to assist with tests.

resolve(*context*)

cobbler.web.templatetags.site.**ifinlist**(*parser*, *token*)
A smarter {% if %} tag for django templates.

While retaining current Django functionality, it also handles equality, greater than and less than operators. Some common case examples:

```
{% if articles|length >= 5 %}...{% endif %}
{% if "ifnotequal tag" != "beautiful" %}...{% endif %}
```

Arguments and operators _must_ have a space between them, so `{% if 1>2 %}` is not a valid smart if tag.

All supported operators are: `or`, `and`, `in`, = (or ==), `!=`, >, >=, < and <=.

cobbler.web.templatetags.site.**listsort**(*value*)

cobbler.web.templatetags.site.**register = <django.template.library.Library object>**
A smarter {% if %} tag for django templates.

While retaining current Django functionality, it also handles equality, greater than and less than operators. Some common case examples:

```
{% if articles|length >= 5 %}...{% endif %}
{% if "ifnotequal tag" != "beautiful" %}...{% endif %}
```

cobbler.web.templatetags.site.**smart_if**(*parser*, *token*)
A smarter {% if %} tag for django templates.

While retaining current Django functionality, it also handles equality, greater than and less than operators. Some common case examples:

```
{% if articles|length >= 5 %}...{% endif %}
{% if "ifnotequal tag" != "beautiful" %}...{% endif %}
```

Arguments and operators _must_ have a space between them, so `{% if 1>2 %}` is not a valid smart if tag.

All supported operators are: `or`, `and`, `in`, = (or ==), `!=`, >, >=, < and <=.

Module contents

Submodules

cobbler.web.field_ui_info module

Describes additional web UI properties of Cobbler fields defined in item_*.py.

Copyright 2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.web.manage module

cobbler.web.settings module

cobbler.web.urls module

cobbler.web.views module

cobbler.web.views.**accept_remote_user**(*request*, *nextsite*)

cobbler.web.views.**aifile_edit**(*request*, *aifile_name=None*, *editmode='edit'*)
This is the page where an automatic OS installation file is edited.

cobbler.web.views.**aifile_list**(*request*, *page=None*)
List all automatic OS installation templates and link to their edit pages.

cobbler.web.views.**aifile_save**(*request*)
This page processes and saves edits to an automatic OS installation file.

cobbler.web.views.**buildiso**(*request*)

cobbler.web.views.**check**(*request*)
Shows a page with the results of 'cobbler check'

cobbler.web.views.**do_login**(*request*)

cobbler.web.views.**do_logout**(*request*)

cobbler.web.views.**error_page**(*request*, *message*)
This page is used to explain error messages to the user.

cobbler.web.views.**eventlog**(*request*, *event=0*)
Shows the log for a given event.

cobbler.web.views.**events**(*request*)
This page presents a list of all the events and links to the event log viewer.

cobbler.web.views.**generic_copy**(*request*, *what*, *obj_name=None*, *obj_newname=None*)
Copies an object.

cobbler.web.views.**generic_delete**(*request*, *what*, *obj_name=None*)
Deletes an object.

cobbler.web.views.**generic_domulti**(*request*, *what*, *multi_mode=None*, *multi_arg=None*)
Process operations like profile reassignment, netboot toggling, and deletion which occur on all items that are checked on the list page.

cobbler.web.views.**generic_edit**(*request*, *what=None*, *obj_name=None*, *editmode='new'*)
Presents an editor page for any type of object. While this is generally standardized, systems are a little bit special.

cobbler.web.views.**generic_rename**(*request*, *what*, *obj_name=None*, *obj_newname=None*)
Renames an object.

cobbler.web.views.**generic_save**(*request*, *what*)
Saves an object back using the Cobbler API after clearing any 'generic_edit' page.

cobbler.web.views.**genlist**(*request*, *what*, *page=None*)
Lists all object types, complete with links to actions on those objects.

cobbler.web.views.**get_fields**(*what*, *is_subobject*, *seed_item=None*)
Helper function. Retrieves the field table from the Cobbler objects and formats it in a way to make it useful for Django templating. The field structure indicates what fields to display and what the default values are, etc.

cobbler.web.views.**get_network_interface_fields**()
Create network interface fields UI metadata based on network interface fields metadata

@return list network interface fields UI metadata

cobbler.web.views.**hardlink**(*request*)
Hardlinks files between repos and install trees to save space.

cobbler.web.views.**import_prompt**(*request*)

cobbler.web.views.**import_run**(*request*)

cobbler.web.views.**index**(*request*)
This is the main greeting page for Cobbler web.

cobbler.web.views.**login**(*request*, *next=None*, *message=None*, *expired=False*)

cobbler.web.views.**modify_list**(*request*, *what*, *pref*, *value=None*)
This function is used in the generic list view to modify the page/column sort/number of items shown per page, and also modify the filters.

This function modifies the session object to store these preferences persistently.

cobbler.web.views.**random_mac**(*request*, *virttype='xenpv'*)
Used in an ajax call to fill in a field with a mac address.

cobbler.web.views.**replicate**(*request*)
Replicate configuration from the central Cobbler server, configured in /etc/cobbler/settings (note: this is uni-directional!)

FIXME: this is disabled because we really need a web page to provide options for this command.

cobbler.web.views.**reposync**(*request*)
Syncs all repos that are configured to be synced.

cobbler.web.views.**setting_edit**(*request*, *setting_name=None*)

cobbler.web.views.**setting_list**(*request*)
This page presents a list of all the settings to the user. They are not editable.

cobbler.web.views.**setting_save**(*request*)

cobbler.web.views.**snippet_edit**(*request*, *snippet_name=None*, *editmode='edit'*)
This page edits a specific snippet.

cobbler.web.views.**snippet_list**(*request*, *page=None*)
This page lists all available snippets and has links to edit them.

cobbler.web.views.**snippet_save**(*request*)
This snippet saves a snippet once edited.

cobbler.web.views.**sync**(*request*)
Runs 'cobbler sync' from the API when the user presses the sync button.

cobbler.web.views.**task_created**(*request*)
Let's the user know what to expect for event updates.

cobbler.web.views.**test_user_authenticated**(*request*)

Module contents

8.2 Submodules

8.3 cobbler.api module

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.api.**CobblerAPI**(*is_cobblerd=False*)
Bases: object

Python API module for Cobbler. See source for cobbler.py, or pydoc, for example usage. Cli apps and daemons should import api.py, and no other Cobbler code.

acl_config(*adduser=None*, *addgroup=None*, *removeuser=None*, *removegroup=None*, *logger=None*)
Configures users/groups to run the Cobbler CLI as non-root. Pass in only one option at a time. Powers `cobbler aclconfig`.

Parameters

- **adduser** –
- **addgroup** –
- **removeuser** –
- **removegroup** –

- **logger** – The logger to audit the removal with.

add_distro(*ref*, *check_for_duplicate_names=False*, *save=True*, *logger=None*)
Add a distribution to Cobbler.

Parameters

- **ref** – The identifier for the object to add to a collection.
- **check_for_duplicate_names** – If the name should be unique or can be present multiple times.
- **save** – If the item should be persisted.
- **logger** – The logger to audit the removal with.

add_file(*ref*, *check_for_duplicate_names=False*, *save=True*, *logger=None*)
Add a file to Cobbler.

Parameters

- **ref** – The identifier for the object to add to a collection.
- **check_for_duplicate_names** – If the name should be unique or can be present multiple times.
- **save** – If the item should be persisted.
- **logger** – The logger to audit the removal with.

add_image(*ref*, *check_for_duplicate_names=False*, *save=True*, *logger=None*)
Add an image to Cobbler.

Parameters

- **ref** – The identifier for the object to add to a collection.
- **check_for_duplicate_names** – If the name should be unique or can be present multiple times.
- **save** – If the item should be persisted.
- **logger** – The logger to audit the removal with.

add_item(*what*, *ref*, *check_for_duplicate_names=False*, *save=True*, *logger=None*)
Add an abstract item to a collection of its specific items. This is not meant for external use. Please reefer to one of the specific methods `add_<type>`.

Parameters

- **what** – The item type.
- **ref** – The identifier for the object to add to a collection.
- **check_for_duplicate_names** – If the name should be unique or can be present multiple times.
- **save** – If the item should be persisted.
- **logger** – The logger to audit the removal with.

add_mgmtclass(*ref*, *check_for_duplicate_names=False*, *save=True*, *logger=None*)
Add a management class to Cobbler.

Parameters

- **ref** – The identifier for the object to add to a collection.
- **check_for_duplicate_names** – If the name should be unique or can be present multiple times.
- **save** – If the item should be persisted.
- **logger** – The logger to audit the removal with.

add_package(*ref*, *check_for_duplicate_names=False*, *save=True*, *logger=None*)
Add a package to Cobbler.

Parameters

- **ref** – The identifier for the object to add to a collection.
- **check_for_duplicate_names** – If the name should be unique or can be present multiple times.
- **save** – If the item should be persisted.
- **logger** – The logger to audit the removal with.

add_profile(*ref*, *check_for_duplicate_names=False*, *save=True*, *logger=None*)
Add a profile to Cobbler.

Parameters

- **ref** – The identifier for the object to add to a collection.
- **check_for_duplicate_names** – If the name should be unique or can be present multiple times.
- **save** – If the item should be persisted.
- **logger** – The logger to audit the removal with.

add_repo(*ref*, *check_for_duplicate_names=False*, *save=True*, *logger=None*)
Add a repository to Cobbler.

Parameters

- **ref** – The identifier for the object to add to a collection.
- **check_for_duplicate_names** – If the name should be unique or can be present multiple times.
- **save** – If the item should be persisted.
- **logger** – The logger to audit the removal with.

add_system(*ref*, *check_for_duplicate_names=False*, *check_for_duplicate_netinfo=False*, *save=True*, *logger=None*)
Add a system to Cobbler.

Parameters

- **ref** – The identifier for the object to add to a collection.
- **check_for_duplicate_names** – If the name should be unique or can be present multiple times.
- **check_for_duplicate_netinfo** – If the name of the network interface should be unique or can be present multiple times.
- **save** – If the item should be persisted.
- **logger** – The logger to audit the removal with.

authenticate(*user*, *password*)
(Remote) access control. This depends on the chosen authentication module. Cobbler internal use only.

Parameters

- **user** – The username to check for authentication.
- **password** – The password to check for authentication.

Returns Whether the action succeeded or not.

authorize(*user*, *resource*, *arg1=None*, *arg2=None*)
(Remote) access control. This depends on the chosen authorization module. Cobbler internal use only.

Parameters

- **user** – The username to check for authorization.
- **resource** – The type of resource which should be checked for access from the supplied user.
- **arg1** – The actual resource to check for authorization.
- **arg2** – Not known what this parameter does exactly.

Returns The return code of the action.

auto_add_repos()
Import any repos this server knows about and mirror them. Run `cobbler reposync` to apply the changes. Credit: Seth Vidal.

build_iso(*iso=None*, *profiles=None*, *systems=None*, *buildisodir=None*, *distro=None*, *standalone=None*, *airgapped=None*, *source=None*, *exclude_dns=None*, *xorrisofs_opts=None*, *logger=None*)
Build an iso image which may be network bootable or not.

Parameters

- **iso** –
- **profiles** –
- **systems** –
- **buildisodir** –
- **distro** –
- **standalone** –
- **airgapped** –
- **source** –
- **exclude_dns** –
- **xorrisofs_opts** –
- **logger** – The logger to audit the removal with.

check(*logger=None*)
See if all preqs for network booting are valid. This returns a list of strings containing instructions on things to correct. An empty list means there is nothing to correct, but that still doesn't mean there are configuration errors. This is mainly useful for human admins, who may, for instance, forget to properly set up their TFTP servers for PXE, etc.

Parameters **logger** – The logger to audit the removal with.

Returns None or a list of things to address.

Return type None or list

clear_logs(*system*, *logger=None*)
Clears console and anamon logs for system

Parameters

- **system** – The system to clear logs of.
- **logger** – The logger to audit the log clearing with.

copy_distro(*ref*, *newname*)
This method copies a distro which is just different in the name of the object.

Parameters

- **ref** – The object itself which gets copied.
- **newname** – The new name of the newly created object.

copy_file(*ref*, *newname*)
This method copies a file which is just different in the name of the object.

Parameters

- **ref** – The object itself which gets copied.
- **newname** – The new name of the newly created object.

copy_image(*ref*, *newname*)
This method copies an image which is just different in the name of the object.

Parameters

- **ref** – The object itself which gets copied.
- **newname** – The new name of the newly created object.

copy_item(*what*, *ref*, *newname*, *logger=None*)
General copy method which is called by the specific methods.

Parameters

- **what** – The collection type which gets copied.
- **ref** – The object itself which gets copied.
- **newname** – The new name of the newly created object.
- **logger** – The logger which is used for auditing the copy task.

copy_mgmtclass(*ref*, *newname*)
This method copies a management class which is just different in the name of the object.

Parameters

- **ref** – The object itself which gets copied.
- **newname** – The new name of the newly created object.

copy_package(*ref*, *newname*)
This method copies a package which is just different in the name of the object.

Parameters

- **ref** – The object itself which gets copied.
- **newname** – The new name of the newly created object.

copy_profile(*ref*, *newname*)
This method copies a profile which is just different in the name of the object.

Parameters

- **ref** – The object itself which gets copied.
- **newname** – The new name of the newly created object.

copy_repo(*ref*, *newname*)
This method copies a repository which is just different in the name of the object.

Parameters

- **ref** – The object itself which gets copied.
- **newname** – The new name of the newly created object.

copy_system(*ref*, *newname*)
This method copies a system which is just different in the name of the object.

Parameters

- **ref** – The object itself which gets copied.
- **newname** – The new name of the newly created object.

deserialize()
Load cobbler_collections from disk. Cobbler internal use only.

distros()
Return the current list of distributions

dlcontent(*force=False*, *logger=None*)
Downloads bootloader content that may not be avialable in packages for the given arch, ex: if installing on PPC, get syslinux. If installing on x86_64, get elilo, etc.

Parameters

- **force** – Force the download, although the content may be already downloaded.
- **logger** – The logger to audit the removal with.

dump_vars(*obj*, *format=False*)
Dump all known variables related to that object.

Parameters

- **obj** – The object for which the variables should be dumped.
- **format** – If True the values will align in one column and be pretty printed for cli example.

Returns A dictionary with all the information which could be collected.

files()
Return the current list of files

find_distro(*name=None*, *return_list=False*, *no_errors=False*, ***kargs*)
Find a distribution via a name or keys specified in the `**kargs`.

Parameters

- **name** – The name to search for.
- **return_list** – If only the first result or all results should be returned.
- **no_errors** – Silence some errors which would raise if this turned to False.
- **kargs** – Additional key-value pairs which may help in finding the desired objects.

Returns A single object or a list of all search results.

find_file(*name=None*, *return_list=False*, *no_errors=False*, ***kargs*)
Find a file via a name or keys specified in the `**kargs`.

Parameters

- **name** – The name to search for.
- **return_list** – If only the first result or all results should be returned.
- **no_errors** – Silence some errors which would raise if this turned to False.
- **kargs** – Additional key-value pairs which may help in finding the desired objects.

Returns A single object or a list of all search results.

find_image(*name=None*, *return_list=False*, *no_errors=False*, ***kargs*)
Find an image via a name or keys specified in the `**kargs`.

Parameters

- **name** – The name to search for.

- **return_list** – If only the first result or all results should be returned.
- **no_errors** – Silence some errors which would raise if this turned to False.
- **kargs** – Additional key-value pairs which may help in finding the desired objects.

Returns A single object or a list of all search results.

find_items(*what*, *criteria=None*)
This is the abstract base method for finding object int the api. It should not be used by external resources. Please reefer to the specific implementations of this method called `find_<object type>`.

Parameters

- **what** – The object type of the item to search for.
- **criteria** – The dictionary with the key-value pairs to find objects with.

Returns The list of items witch match the search criteria.

find_mgmtclass(*name=None*, *return_list=False*, *no_errors=False*, ***kargs*)
Find a management class via a name or keys specified in the `**kargs`.

Parameters

- **name** – The name to search for.
- **return_list** – If only the first result or all results should be returned.
- **no_errors** – Silence some errors which would raise if this turned to False.
- **kargs** – Additional key-value pairs which may help in finding the desired objects.

Returns A single object or a list of all search results.

find_package(*name=None*, *return_list=False*, *no_errors=False*, ***kargs*)
Find a package via a name or keys specified in the `**kargs`.

Parameters

- **name** – The name to search for.
- **return_list** – If only the first result or all results should be returned.
- **no_errors** – Silence some errors which would raise if this turned to False.
- **kargs** – Additional key-value pairs which may help in finding the desired objects.

Returns A single object or a list of all search results.

find_profile(*name=None*, *return_list=False*, *no_errors=False*, ***kargs*)
Find a profile via a name or keys specified in the `**kargs`.

Parameters

- **name** – The name to search for.
- **return_list** – If only the first result or all results should be returned.
- **no_errors** – Silence some errors which would raise if this turned to False.
- **kargs** – Additional key-value pairs which may help in finding the desired objects.

Returns A single object or a list of all search results.

find_repo(*name=None*, *return_list=False*, *no_errors=False*, ***kargs*)
Find a repository via a name or keys specified in the `**kargs`.

Parameters

- **name** – The name to search for.
- **return_list** – If only the first result or all results should be returned.

- **no_errors** – Silence some errors which would raise if this turned to False.
- **kargs** – Additional key-value pairs which may help in finding the desired objects.

Returns A single object or a list of all search results.

find_system(*name=None*, *return_list=False*, *no_errors=False*, ***kargs*)
Find a system via a name or keys specified in the `**kargs`.

Parameters

- **name** – The name to search for.
- **return_list** – If only the first result or all results should be returned.
- **no_errors** – Silence some errors which would raise if this turned to False.
- **kargs** – Additional key-value pairs which may help in finding the desired objects.

Returns A single object or a list of all search results.

generate_bootcfg(*profile*, *system*)
Generate a boot configuration. The system wins over the profile.

Parameters

- **profile** – The profile to return the configuration for.
- **system** – The system to return the configuration for.

Returns The generated configuration file.

generate_gpxe(*profile*, *system*)
Generate the gpxe configuration files. The system wins over the profile.

Parameters

- **profile** – The profile to return the configuration for.
- **system** – The system to return the configuration for.

Returns The generated configuration file.

generate_script(*profile*, *system*, *name*)
Generate an autoinstall script for the specified profile or system. The system wins over the profile.

Parameters

- **profile** – The profile to generate the script for.
- **system** – The system to generate the script for.
- **name** – The name of the script which should be generated.

Returns The generated script or an error message.

get_distros_since(*mtime*, *collapse=False*)
Returns distros modified since a certain time (in seconds since Epoch)

Parameters

- **mtime** – The timestamp which marks the gate if an object is included or not.
- **collapse** – collapse=True specifies returning a dict instead of objects.

Returns The list of distros which are newer then the given timestamp.

get_files_since(*mtime*, *collapse=False*)
Return files modified since a certain time (in seconds since Epoch)

Parameters

- **mtime** – The timestamp which marks the gate if an object is included or not.

- **collapse** – If True then this specifies that a list of dicts should be returned instead of a list of objects.

Returns The list of files which are newer then the given timestamp.

Return type list

get_images_since(*mtime*, *collapse=False*)
Return images modified since a certain time (in seconds since Epoch)

Parameters

- **mtime** – The timestamp which marks the gate if an object is included or not.
- **collapse** – If True then this specifies that a list of dicts should be returned instead of a list of objects.

Returns The list of images which are newer then the given timestamp.

Return type list

get_item(*what*, *name*)
Get a general item.

Parameters

- **what** (*str*) – The item type to retrieve from the internal database.
- **name** (*str*) – The name of the item to retrieve.

Returns An item of the desired type.

get_items(*what*)
Get all items of a collection.

Parameters **what** (*str*) – The collection to query.

Returns The items which were queried. May return no items.

get_mgmtclasses_since(*mtime*, *collapse=False*)
Return management classes modified since a certain time (in seconds since Epoch)

Parameters

- **mtime** – The timestamp which marks the gate if an object is included or not.
- **collapse** – If True then this specifies that a list of dicts should be returned instead of a list of objects.

Returns The list of management classes which are newer then the given timestamp.

Return type list

get_module_by_name(*module_name*)
Returns a loaded Cobbler module named 'name', if one exists, else None. Cobbler internal use only.

Parameters **module_name** –

Returns

get_module_from_file(*section*, *name*, *fallback=None*)
Looks in `/etc/cobbler/modules.conf` for a section called 'section' and a key called 'name', and then returns the module that corresponds to the value of that key. Cobbler internal use only.

Parameters

- **section** –
- **name** –
- **fallback** –

Returns

get_module_name_from_file(*section*, *name*, *fallback=None*)
Looks up a module the same as `get_module_from_file` but returns the module name rather than the module itself.

Parameters

- **section** –
- **name** –
- **fallback** –

Returns

get_modules_in_category(*category*)
Returns all modules in a given category, for instance "serializer", or "cli". Cobbler internal use only.

Parameters **category** – The category to check.

Returns The list of modules.

get_packages_since(*mtime*, *collapse=False*)
Return packages modified since a certain time (in seconds since Epoch)

Parameters

- **mtime** – The timestamp which marks the gate if an object is included or not.
- **collapse** – If True then this specifies that a list of dicts should be returned instead of a list of objects.

Returns The list of packages which are newer then the given timestamp.

Return type list

get_profiles_since(*mtime*, *collapse=False*)
Returns profiles modified since a certain time (in seconds since Epoch)

Parameters

- **mtime** – The timestamp which marks the gate if an object is included or not.
- **collapse** – If True then this specifies that a list of dicts should be returned instead of a list of objects.

Returns The list of profiles which are newer then the given timestamp.

Return type list

get_repo_config_for_profile(*obj*)
Get the repository configuration for the specified profile

Parameters **obj** – The profile to return the configuration for.

Returns The repository configuration as a string.

Return type str

get_repo_config_for_system(*obj*)
Get the repository configuration for the specified system.

Parameters **obj** – The system to return the configuration for.

Returns The repository configuration as a string.

Return type str

get_repos_since(*mtime*, *collapse=False*)
Return repositories modified since a certain time (in seconds since Epoch)

Parameters

- **mtime** – The timestamp which marks the gate if an object is included or not.

- **collapse** – If True then this specifies that a list of dicts should be returned instead of a list of objects.

Returns The list of repositories which are newer then the given timestamp.

Return type list

get_signatures()
This returns the local signature cache.

Returns The dict containing all signatures.

get_sync(*verbose=False*, *logger=None*)
Get a Cobbler Sync object which may be executed through the call of `obj.run()`.

Parameters

- **verbose** – If the action should be just logged as needed or (if True) as much verbose as possible.
- **logger** – The logger to audit the removal with.

Returns An instance of the CobblerSync class to execute the sync with.

get_systems_since(*mtime*, *collapse=False*)
Return systems modified since a certain time (in seconds since Epoch)

Parameters

- **mtime** – The timestamp which marks the gate if an object is included or not.
- **collapse** – If True then this specifies that a list of dicts should be returned instead of a list of objects.

Returns The list of systems which are newer then the given timestamp.

Return type list

get_template_file_for_profile(*obj*, *path*)
Get the template for the specified profile.

Parameters

- **obj** – The object which is related to that template.
- **path** – The path to the template.

Returns The template as in its string representation.

get_template_file_for_system(*obj*, *path*)
Get the template for the specified system.

Parameters

- **obj** – The object which is related to that template.
- **path** – The path to the template.

Returns The template as in its string representation.

hardlink(*logger=None*)
Hardlink all files where this is possible to improve performance.

Parameters **logger** – The logger to audit the removal with.

Returns The return code of the subprocess call which actually hardlinks the files.

images()
Return the current list of images

import_tree(*mirror_url*, *mirror_name*, *network_root=None*, *autoinstall_file=None*, *rsync_flags=None*, *arch=None*, *breed=None*, *os_version=None*, *logger=None*)
Automatically import a directory tree full of distribution files.

Parameters

- **mirror_url** – Can be a string that represents a path, a user@host syntax for SSH, or an rsync:// address. If mirror_url is a filesystem path and mirroring is not desired, set network_root to something like "nfs://path/to/mirror_url/root"
- **mirror_name** –
- **network_root** –
- **autoinstall_file** –
- **rsync_flags** –
- **arch** –
- **breed** –
- **os_version** –
- **logger** – The logger to audit the removal with.

is_selinux_enabled()
Returns whether selinux is enabled on the Cobbler server. We check this just once at Cobbler API init time, because a restart is required to change this; this does /not/ check enforce/permissive, nor does it need to. :rtype: bool

is_selinux_supported()
Returns whether or not the OS is sufficient enough to run with SELinux enabled (currently EL 5 or later).

Returns False per default. If Distro is Redhat and Version >= 5 then it returns true.

Return type bool

last_modified_time()
Returns the time of the last modification to Cobbler, made by any API instance, regardless of the serializer type.

Returns 0 if there is no file where the information required for this method is saved.

Return type float

log(*msg*, *args=None*, *debug=False*)
Logs a message with the already initiated logger of this object.

Parameters

- **msg** (*str*) – The message to log.
- **args** – Optional message which gets appended to the main msg with a ';'.
- **debug** (*bool*) – Weather the logged message is a debug message (true) or info (false).

mgmtclasses()
Return the current list of mgmtclasses

new_distro(*is_subobject=False*)
Returns a new empty distro object. This distro is not automatically persited. Persistance is achieved via `save()`.

Parameters **is_subobject** (*bool*) – If the object created is a subobject or not.

Returns An empty Distro object.

new_file(*is_subobject=False*)
Returns a new empty file object. This file is not automatically persisted. Persistence is achieved via `save()`.

Parameters **is_subobject** (*bool*) – If the object created is a subobject or not.

Returns An empty File object.

new_image(*is_subobject=False*)
Returns a new empty image object. This image is not automatically persisted. Persistence is achieved via `save()`.

Parameters **is_subobject** (*bool*) – If the object created is a subobject or not.

Returns An empty image object.

new_mgmtclass(*is_subobject=False*)
Returns a new empty mgmtclass object. This mgmtclass is not automatically persisted. Persistence is achieved via `save()`.

Parameters **is_subobject** (*bool*) – If the object created is a subobject or not.

Returns An empty mgmtclass object.

new_package(*is_subobject=False*)
Returns a new empty package object. This package is not automatically persisted. Persistence is achieved via `save()`.

Parameters **is_subobject** (*bool*) – If the object created is a subobject or not.

Returns An empty Package object.

new_profile(*is_subobject=False*)
Returns a new empty profile object. This profile is not automatically persisted. Persistence is achieved via `save()`.

Parameters **is_subobject** (*bool*) – If the object created is a subobject or not.

Returns An empty Profile object.

new_repo(*is_subobject=False*)
Returns a new empty repo object. This repository is not automatically persisted. Persistence is achieved via `save()`.

Parameters **is_subobject** (*bool*) – If the object created is a subobject or not.

Returns An empty repo object.

new_system(*is_subobject=False*)
Returns a new empty system object. This system is not automatically persisted. Persistence is achieved via `save()`.

Parameters **is_subobject** (*bool*) – If the object created is a subobject or not.

Returns An empty System object.

packages()
Return the current list of packages

power_system(*system*, *power_operation*, *user=None*, *password=None*, *logger=None*)
Power on / power off / get power status /reboot a system.

Parameters

- **system** (*str*) – Cobbler system
- **power_operation** (*str*) – power operation. Valid values: on, off, reboot, status
- **user** (*str*) – power management user
- **password** (*str*) – power management password
- **logger** (*Logger*) – The logger to audit the removal with.

Returns bool if operation was successful

profiles()
Return the current list of profiles

remove_distro(*ref*, *recursive=False*, *delete=True*, *with_triggers=True*, *logger=None*)
Remove a distribution from Cobbler.

Parameters

- **ref** – The internal unique handle for the item.
- **recursive** – If the item should recursively should delete dependencies on itself.
- **delete** – Not known what this parameter does exactly.
- **with_triggers** – Whether you would like to have the removal triggers executed or not.
- **logger** – The logger to audit the removal with.

remove_file(*ref*, *recursive=False*, *delete=True*, *with_triggers=True*, *logger=None*)
Remove a file from Cobbler.

Parameters

- **ref** – The internal unique handle for the item.
- **recursive** – If the item should recursively should delete dependencies on itself.
- **delete** – Not known what this parameter does exactly.
- **with_triggers** – Whether you would like to have the removal triggers executed or not.
- **logger** – The logger to audit the removal with.

remove_image(*ref*, *recursive=False*, *delete=True*, *with_triggers=True*, *logger=None*)
Remove a image from Cobbler.

Parameters

- **ref** – The internal unique handle for the item.
- **recursive** – If the item should recursively should delete dependencies on itself.
- **delete** – Not known what this parameter does exactly.
- **with_triggers** – Whether you would like to have the removal triggers executed or not.
- **logger** – The logger to audit the removal with.

remove_item(*what*, *ref*, *recursive=False*, *delete=True*, *with_triggers=True*, *logger=None*)
Remove a general item. This method should not be used by an external api. Please use the specific remove_<itemtype> methods.

Parameters

- **what** – The type of the item.
- **ref** – The internal unique handle for the item.
- **recursive** – If the item should recursively should delete dependencies on itself.
- **delete** – Not known what this parameter does exactly.
- **with_triggers** – Whether you would like to have the removal triggers executed or not.
- **logger** – The logger to audit the removal with.

remove_mgmtclass(*ref*, *recursive=False*, *delete=True*, *with_triggers=True*, *logger=None*)
Remove a management class from Cobbler.

Parameters

- **ref** – The internal unique handle for the item.

- **recursive** – If the item should recursively should delete dependencies on itself.
- **delete** – Not known what this parameter does exactly.
- **with_triggers** – Whether you would like to have the removal triggers executed or not.
- **logger** – The logger to audit the removal with.

remove_package(*ref*, *recursive=False*, *delete=True*, *with_triggers=True*, *logger=None*)
Remove a package from Cobbler.

Parameters

- **ref** – The internal unique handle for the item.
- **recursive** – If the item should recursively should delete dependencies on itself.
- **delete** – Not known what this parameter does exactly.
- **with_triggers** – Whether you would like to have the removal triggers executed or not.
- **logger** – The logger to audit the removal with.

remove_profile(*ref*, *recursive=False*, *delete=True*, *with_triggers=True*, *logger=None*)
Remove a profile from Cobbler.

Parameters

- **ref** – The internal unique handle for the item.
- **recursive** – If the item should recursively should delete dependencies on itself.
- **delete** – Not known what this parameter does exactly.
- **with_triggers** – Whether you would like to have the removal triggers executed or not.
- **logger** – The logger to audit the removal with.

remove_repo(*ref*, *recursive=False*, *delete=True*, *with_triggers=True*, *logger=None*)
Remove a repository from Cobbler.

Parameters

- **ref** – The internal unique handle for the item.
- **recursive** – If the item should recursively should delete dependencies on itself.
- **delete** – Not known what this parameter does exactly.
- **with_triggers** – Whether you would like to have the removal triggers executed or not.
- **logger** – The logger to audit the removal with.

remove_system(*ref*, *recursive=False*, *delete=True*, *with_triggers=True*, *logger=None*)
Remove a system from Cobbler.

Parameters

- **ref** – The internal unique handle for the item.
- **recursive** – If the item should recursively should delete dependencies on itself.
- **delete** – Not known what this parameter does exactly.
- **with_triggers** – Whether you would like to have the removal triggers executed or not.
- **logger** – The logger to audit the removal with.

rename_distro(*ref*, *newname*, *logger=None*)
Rename a distro to a new name.

Parameters

- **ref** – The internal unique handle for the item.
- **newname** – The new name for the item.
- **logger** – The logger to audit the removal with.

rename_file(*ref*, *newname*, *logger=None*)
Rename a file to a new name.

Parameters

- **ref** – The internal unique handle for the item.
- **newname** – The new name for the item.
- **logger** – The logger to audit the removal with.

rename_image(*ref*, *newname*, *logger=None*)
Rename an image to a new name.

Parameters

- **ref** – The internal unique handle for the item.
- **newname** – The new name for the item.
- **logger** – The logger to audit the removal with.

rename_item(*what*, *ref*, *newname*, *logger=None*)
Remove a general item. This method should not be used by an external api. Please use the specific rename_<itemtype> methods.

Parameters

- **what** – The type of object which should be renamed.
- **ref** – The internal unique handle for the item.
- **newname** – The new name for the item.
- **logger** – The logger to audit the removal with.

rename_mgmtclass(*ref*, *newname*, *logger=None*)
Rename a management class to a new name.

Parameters

- **ref** – The internal unique handle for the item.
- **newname** – The new name for the item.
- **logger** – The logger to audit the removal with.

rename_package(*ref*, *newname*, *logger=None*)
Rename a package to a new name.

Parameters

- **ref** – The internal unique handle for the item.
- **newname** – The new name for the item.
- **logger** – The logger to audit the removal with.

rename_profile(*ref*, *newname*, *logger=None*)
Rename a profile to a new name.

Parameters

- **ref** – The internal unique handle for the item.

- **newname** – The new name for the item.
- **logger** – The logger to audit the removal with.

rename_repo(*ref*, *newname*, *logger=None*)
Rename a repository to a new name.

Parameters

- **ref** – The internal unique handle for the item.
- **newname** – The new name for the item.
- **logger** – The logger to audit the removal with.

rename_system(*ref*, *newname*, *logger=None*)
Rename a system to a new name.

Parameters

- **ref** – The internal unique handle for the item.
- **newname** – The new name for the item.
- **logger** – The logger to audit the removal with.

replicate(*cobbler_master=None*, *port='80'*, *distro_patterns=''*, *profile_patterns=''*, *system_patterns=''*, *repo_patterns=''*, *image_patterns=''*, *mgmtclass_patterns=None*, *package_patterns=None*, *file_patterns=None*, *prune=False*, *omit_data=False*, *sync_all=False*, *use_ssl=False*, *logger=None*)
Pull down data/configs from a remote Cobbler server that is a master to this server.

Parameters

- **cobbler_master** (*str*) – The hostname/URL of the other Cobbler server
- **port** (*str*) – The port to use for the replication task.
- **distro_patterns** – The pattern of distros which should be synced.
- **profile_patterns** – The pattern of profiles which should be synced.
- **system_patterns** – The pattern of systems which should be synced.
- **repo_patterns** – The pattern of repositories which should be synced.
- **image_patterns** – The pattern of images which should be synced.
- **mgmtclass_patterns** – The pattern of management classes which should be synced.
- **package_patterns** – The pattern of packages which should be synced.
- **file_patterns** – The pattern of files which should be synced.
- **prune** (*bool*) – Whether the object not on the master should be removed or not.
- **omit_data** (*bool*) – If the data downloaded by the current Cobbler server should be rsynced to the destination server.
- **sync_all** (*bool*) – This parameter behaves similarly to a dry run argument. If True then everything will executed, if False then only some things are synced.
- **use_ssl** (*bool*) – Whether SSL should be used (True) or not (False).
- **logger** – The logger to audit the removal with.

report(*report_what=None*, *report_name=None*, *report_type=None*, *report_fields=None*, *report_noheaders=None*)
Report functionality for Cobbler.

Parameters

- **report_what** – The object type that should be reported.

- **report_name** – The name of the object which should be possibly reported.
- **report_type** – May be either “text”, “csv”, “mediawiki”, “trac” or “doku”.
- **report_fields** – Specify “all” or the fields you want to be reported.
- **report_noheaders** – If the column headers should be included in the output or not.

repos()
Return the current list of repos

reposync(*name=None*, *tries=1*, *nofail=False*, *logger=None*)
Take the contents of `/var/lib/cobbler/repos` and update them – or create the initial copy if no contents exist yet.

Parameters

- **name** – The name of the repository to run reposync for.
- **tries** – How many tries should be executed before the action fails.
- **nofail** – If True then the action will fail, otherwise the action will just be skipped. This respects the `tries` parameter.
- **logger** – The logger to audit the removal with.

serialize()
Save the cobbler_collections to disk. Cobbler internal use only.

settings()
Return the application configuration

signature_update(*logger*)
Update all signatures from the URL specified in the settings.

Parameters **logger** – The logger to audit the removal with.

status(*mode*, *logger=None*)
Get the status of the current Cobbler instance.

Parameters

- **mode** – “text” or anything else. Meaning whether the output is thought for the terminal or not.
- **logger** – The logger to audit the removal with.

Returns The current status of Cobbler.

sync(*verbose=False*, *logger=None*)
Take the values currently written to the configuration files in /etc, and /var, and build out the information tree found in /tftpboot. Any operations done in the API that have not been saved with serialize() will NOT be synchronized with this command.

Parameters

- **verbose** – If the action should be just logged as needed or (if True) as much verbose as possible.
- **logger** – The logger to audit the removal with.

sync_dhcp(*verbose=False*, *logger=None*)
Only build out the DHCP configuration

Parameters

- **verbose** – If the action should be just logged as needed or (if True) as much verbose as possible.
- **logger** – The logger to audit the removal with.

systems()
Return the current list of systems

validate_autoinstall_files(*logger=None*)
Validate if any of the autoinstallation files are invalid and if yes report this.

Parameters **logger** – The logger to audit the removal with.

version(*extended=False*)
What version is Cobbler?

If extended == False, returns a float for backwards compatibility If extended == True, returns a dict:

gitstamp – the last git commit hash gitdate – the last git commit date on the builder machine builddate – the time of the build version – something like "1.3.2" version_tuple – something like [1, 3, 2]

Parameters **extended** (*bool*) – False returns a float, True a Dictionary.

8.4 cobbler.autoinstall_manager module

class cobbler.autoinstall_manager.**AutoInstallationManager**(*collection_mgr*, *logger=None*)

Bases: object

Manage automatic installation templates, snippets and final files

generate_autoinstall(*profile=None*, *system=None*)
Generates the autoinstallation for a system or a profile. You may only specifify one parameter. If you specify both, the system is generated and the profile argument is ignored.

Parameters

- **profile** – The Cobbler profile you want an autoinstallation generated for.
- **system** – The Cobbler system you want an autoinstallation generated for.

Returns The rendered template for the system or profile.

Return type str

get_autoinstall_snippets()
Get a list of all autoinstallation snippets.

Returns The list of snippets

Return type list

get_autoinstall_templates()
Get automatic OS installation templates

Returns A list of automatic installation templates

Return type list

is_autoinstall_in_use(*name*)
Reports the status if a given system is currently being provisioned.

Parameters **name** (*str*) – The name of the system.

Returns Whether the system is in install mode or not.

Return type bool

log_autoinstall_validation_errors(*errors_type*, *errors*)
Log automatic installation file errors

Parameters

- **errors_type** (*int*) – validation errors type
- **errors** (*list*) – A list with all the errors which occurred.

read_autoinstall_snippet(*file_path*)
Reads a autoinstall snippet from underneath the configured snippet base dir.

Parameters **file_path** (*str*) – The relative file path under the configured snippets base dir.

Returns The read snippet.

Return type str

read_autoinstall_template(*file_path*)
Read an automatic OS installation template

Parameters **file_path** (*str*) – automatic installation template relative file path

Returns automatic installation template content

Return type str

remove_autoinstall_snippet(*file_path*)
Remove the autoinstall snippet with the given path.

Parameters **file_path** (*str*) – The path relative to the configured snippet root.

Returns A boolean indicating the success of the task.

Return type bool

remove_autoinstall_template(*file_path*)
Remove an automatic OS installation template

Parameters **file_path** (*str*) – automatic installation template relative file path

validate_autoinstall_file(*obj*, *is_profile*)
Validate automatic installation file used by a system/profile.

Parameters

- **obj** – system/profile
- **is_profile** (*bool*) – if obj is a profile

Returns [bool, int, list] list with validation result, errors type and list of errors

Return type list

validate_autoinstall_files(*logger=None*)
Determine if Cobbler automatic OS installation files will be accepted by corresponding Linux distribution installers. The presence of an error does not imply that the automatic installation file is bad, only that the possibility exists. Automatic installation file validators are not available for all automatic installation file types and on all operating systems in which Cobbler may be installed.

Parameters **logger** – The logger which watches the validation

Returns True if all automatic installation files are valid, otherwise false.

Return type bool

validate_autoinstall_snippet_file_path(*snippet*, *new_snippet=False*)
Validate the snippet's relative file path.

Parameters

- **snippet** (*str*) – automatic installation snippet relative file path
- **new_snippet** (*bool*) – when set to true new filenames are allowed

Returns Snippet if successful otherwise raises an exception.

Raises ***CX*** – Raised when the arguments are invalid or the action performed raised an internal error.

Return type str

validate_autoinstall_template_file_path(*autoinstall*, *for_item=True*, *new_autoinstall=False*)
Validate the automatic installation template's relative file path.

Parameters

- **autoinstall** (*str*) – automatic installation template relative file path
- **for_item** (*bool*) – enable/disable special handling for Item objects
- **new_autoinstall** (*bool*) – when set to true new filenames are allowed

Returns automatic installation template relative file path

Return type str

write_autoinstall_snippet(*file_path*, *data*)
Writes a snippet with the given content to the relative path under the snippet root directory.

Parameters

- **file_path** (*str*) – The relative path under the configured snippet base dir.
- **data** (*str*) – The snippet code.

write_autoinstall_template(*file_path*, *data*)
Write an automatic OS installation template

Parameters

- **file_path** (*str*) – automatic installation template relative file path
- **data** (*str*) – automatic installation template content

Return type bool

8.5 cobbler.autoinstallgen module

Builds out filesystem trees/data based on the object tree. This is the code behind 'cobbler sync'.

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.autoinstallgen.**AutoInstallationGen**(*collection_mgr*)
Bases: object

Handles conversion of internal state to the tftpboot tree layout

addAutoYaSTScript(*document*, *type*, *source*)
Add scripts to an existing AutoYaST XML.

Parameters

- **document** – The existing AutoYaST XML object.

- **type** – The type of the script which should be added.
- **source** – The source of the script. This should be ideally a string.

createAutoYaSTScript(*document*, *script*, *name*)
This method attaches a script with a given name to an existing AutoYaST XML file.

Parameters

- **document** – The existing AutoYaST XML file.
- **script** – The script to attach.
- **name** – The name of the script.

Returns The AutoYaST file with the attached script.

generate_autoinstall(*profile=None*, *system=None*)
This is an internal method for generating an autoinstall config/script. Please use the `generate_autoinstall_for_*` methods. If you insist on using this mehtod please only supply a profile or a system, not both.

Parameters

- **profile** – The profile to use for generating the autoinstall config/script.
- **system** – The system to use for generating the autoinstall config/script. If both arguments are given, this wins.

Returns The autoinstall script or configuration file as a string.

Return type str

generate_autoinstall_for_profile(*g*)
Generate an autoinstall config or script for a profile.

Parameters **g** – The Profile to generate the script/config for.

Returns The generated output or an error message with a human readable description.

Return type str

generate_autoinstall_for_system(*sys_name*)
Generate an autoinstall config or script for a system.

Parameters **sys_name** – The system name to generate an autoinstall script for.

Returns The generated output or an error message with a human readable description.

Return type str

generate_autoyast(*profile=None*, *system=None*, *raw_data=None*)
Generate auto installation information for SUSE distribution (AutoYaST XML file) for a specific system or general profile. Only a system OR profile can be supplied, NOT both.

Parameters

- **profile** – The profile to generate the AutoYaST file for.
- **system** – The system to generate the AutoYaST file for.
- **raw_data** – The raw data which should be included in the profile.

Returns The generated AutoYaST XML file.

Return type str

generate_config_stanza(*obj*, *is_profile=True*)
Add in automatic to configure /etc/yum.repos.d on the remote system if the automatic installation file (template file) contains the magic $yum_config_stanza.

Parameters

- **obj** – The profile or system to generate a generate a config stanza for.

- **is_profile** – If the object is a profile. If False it is assumed that the object is a system.

Returns The curl command to execute to get the configuration for a system or profile.

generate_repo_stanza(*obj*, *is_profile=True*)
Automatically attaches yum repos to profiles/systems in automatic installation files (template files) that contain the magic $yum_repo_stanza variable. This includes repo objects as well as the yum repos that are part of split tree installs, whose data is stored with the distro (example: RHEL5 imports)

Parameters

- **obj** – The profile or system to generate the repo stanza for.
- **is_profile** – If True then obj is a profile, otherwise obj has to be a system. Otherwise this method will silently fail.

Returns The string with the attached yum repos.

Return type str

get_last_errors()
Returns the list of errors generated by the last template render action.

Returns The list of error messages which are available. This may not only contain error messages related to generating autoinstallation configuration and scripts.

Return type list

8.6 cobbler.cexceptions module

Custom exceptions for Cobbler

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

exception cobbler.cexceptions.**CX**(*value*, **args*)
Bases: *cobbler.cexceptions.CobblerException*

This is a general exception which get's thrown often inside Cobbler.

exception cobbler.cexceptions.**CobblerException**(*value*, **args*)
Bases: Exception

This is the default Cobbler exception where all other exceptions are inheriting from.

exception cobbler.cexceptions.**FileNotFoundException**(*value*, **args*)
Bases: *cobbler.cexceptions.CobblerException*

This means that the required file was not found during the process of opening it.

exception cobbler.cexceptions.**NotImplementedException**(*value*, **args*)
Bases: *cobbler.cexceptions.CobblerException*

On the command line interface not everything is always implemented. This is the exception which stated this.

8.7 cobbler.cli module

Command line interface for Cobbler.

class cobbler.cli.**CobblerCLI**(*cliargs*)

Bases: object

check_setup()

Detect permissions and service accessibility problems and provide nicer error messages for them.

cleanup_fault_string(*str*)

Make a remote exception nicely readable by humans so it's not evident that is a remote fault. Users should not have to understand tracebacks.

Parameters **str** – The stacktrace to niceify.

Returns A nicer error messsage.

Return type str

direct_command(*action_name*)

Process non-object based commands like "sync" and "hardlink".

Parameters **action_name** – The action to execute.

Returns Depending on the action.

follow_task(*task_id*)

Follow a task which is remotely executed on the Cobbler-server.

Parameters **task_id** – The id of the task to follow.

get_direct_action(*object_type*, *args*)

If this is a general command, e.g. "cobbler hardlink", return the action, like "hardlink"

Parameters

- **object_type** – Must be None or None is returned.
- **args** – The arg from the CLI.

Returns The action key, "version" or None.

Return type None or strs

get_fields(*object_type*)

For a given name of an object type, return the FIELDS data structure.

Parameters **object_type** – The object to return the fields of.

Returns The fields or None

Return type None or list

get_object_action(*object_type*, *args*)

If this is a CLI command about an object type, e.g. "cobbler distro add", return the action, like "add"

Parameters

- **object_type** – The object type.
- **args** – The args from the CLI.

Returns The action or None.

Return type None or str

get_object_type(*args*)
If this is a CLI command about an object type, e.g. "cobbler distro add", return the type, like "distro"

Parameters **args** – The args from the CLI.

Returns The object type or None

Return type None or str

object_command(*object_type*, *object_action*)
Process object-based commands such as "distro add" or "profile rename"

Parameters

- **object_type** – The object type to execute an action for.
- **object_action** – The action to execute.

Returns Depending on the object and action.

print_help()
Prints general-top level help, e.g. "cobbler –help" or "cobbler" or "cobbler command-does-not-exist"

print_object_help(*object_type*)
Prints the subcommands for a given object, e.g. "cobbler distro –help"

Parameters **object_type** – The object type to print the help for.

print_task(*task_id*)
Pretty print a task executed on the server. This prints to stdout.

Parameters **task_id** – The id of the task to be pretty printed.

run(*args*)
Process the command line and do what the user asks.

Parameters **args** – The args of the CLI

start_task(*name*, *options*)
Start an asynchronous task in the background.

Parameters

- **name** (*str*) – "background_" % name function must exist in remote.py. This function will be called in a subthread.
- **options** (*dict*) – Dictionary of options passed to the newly started thread

Returns Id of the newly started task

Return type str

cobbler.cli.**add_options_from_fields**(*object_type*, *parser*, *fields*, *network_interface_fields*, *settings*, *object_action*)
Add options to the command line from the fields queried from the Cobbler server.

Parameters

- **object_type** – The object type to add options for.
- **parser** – The optparse instance to add options to.
- **fields** – The list of fields to add options for.
- **network_interface_fields** – The list of network interface fields if the object type is a system.

- **settings** – Global cobbler settings as returned from `CollectionManager.settings()`
- **object_action** – The object action to add options for. May be "add", "edit", "find", "copy", "rename", "remove". If none of these options is given then this method does nothing.

`cobbler.cli.list_items(remote, otype)`
List all items of a given object type and print it to stdout.

Parameters

- **remote** – The remote to use as the query-source.
- **otype** – The object type to query.

`cobbler.cli.main()`
CLI entry point

`cobbler.cli.n2s(data)`
Return spaces for None

Parameters **data** – The data to check for.

Returns The data itself or an empty string.

`cobbler.cli.opt(options, k, defval="")`
Returns an option from an Optparse values instance

Parameters

- **options** – The options object to search in.
- **k** – The key which is in the optparse values instance.
- **defval** – The default value to return.

Returns The value for the specified key.

`cobbler.cli.report_item(remote, otype, item=None, name=None)`
Return a single item in a given collection. Either this is an item object or this method searches for a name.

Parameters

- **remote** – The remote to use as the query-source.
- **otype** – The object type to query.
- **item** – The item to display
- **name** – The name to search for and display.

`cobbler.cli.report_items(remote, otype)`
Return all items for a given collection.

Parameters

- **remote** – The remote to use as the query-source. The remote to use as the query-source.
- **otype** – The object type to query.

8.8 cobbler.clogger module

Python standard logging doesn't super-intelligent and won't expose filehandles, which we want. So we're not using it.

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.clogger.**Logger**(*logfile=None*)

Bases: object

Logger class for Cobbler which is wrapped around the Python3 standard logger.

Please don't use this. Utilize the standard logger from Python3 so we can get rid of this eventually.

critical(*msg*)

A critical message which is related to a problem which will halt Cobbler.

Parameters **msg** (*str*) – The message to be logged.

debug(*msg*)

A message which is useful for finding errors or performance problems. Should not be visible in the production usage of Cobbler.

Parameters **msg** (*str*) – The message to be logged.

error(*msg*)

An error message which means that Cobbler will not halt but the future actions may not be executed correctly.

Parameters **msg** (*str*) – The message to be logged.

flat(*msg*)

This uses the print function from the std library. Avoid using this. This is only used for the report command in `cobbler/actions/report.py`

Parameters **msg** (*str*) – The message to be logged.

info(*msg*)

An informational message which should be written to the target log.

Parameters **msg** (*str*) – The message to be logged.

warning(*msg*)

A warning message which could possibly indicate performance or functional problems.

Parameters **msg** (*str*) – The message to be logged.

8.9 cobbler.cobblerd module

Cobbler daemon for logging remote syslog traffic during automatic installation

Copyright 2007-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.cobblerd.**core**(*api*)
Starts Cobbler.

Parameters **api** – The cobbler_api instance which is used for this method.

cobbler.cobblerd.**do_xmlrpc_rw**(*cobbler_api*, *settings*, *port*)
This trys to bring up the Cobbler xmlrpc_api and restart it if it fails.

Parameters

- **cobbler_api** – The cobbler_api instance which is used for this method.
- **settings** – The Cobbler settings instance which is used for this method.
- **port** – The port where the xmlrpc api should run on.

cobbler.cobblerd.**do_xmlrpc_tasks**(*cobbler_api*, *settings*, *xmlrpc_port*)
This trys to bring up the Cobbler xmlrpc_api and restart it if it fails. Tailcall to `do_xmlrpc_rw`.

Parameters

- **cobbler_api** – The cobbler_api instance which is used for this method.
- **settings** – The Cobbler settings instance which is used for this method.
- **xmlrpc_port** – The port where xmlrpc should run on.

cobbler.cobblerd.**log**(*logger*, *msg*)
This logs something with the Cobbler Logger.

Parameters

- **logger** – If this is not none then an info message is printed to the log target. In any other case stderr is used.
- **msg** (*str*) – The message to be logged.

cobbler.cobblerd.**regen_ss_file**()
This is only used for Kerberos auth at the moment. It identifies XMLRPC requests from Apache that have already been cleared by Kerberos.

Returns 1 if this was successful.

8.10 cobbler.configgen module

configgen.py: Generate configuration data.

Copyright 2010 Kelsey Hightower Kelsey Hightower <kelsey.hightower@gmail.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

module for generating configuration manifest using autoinstall_meta data, mgmtclasses, resources, and templates for a given system (hostname)

class cobbler.configgen.**ConfigGen**(*hostname*)
Bases: object

Generate configuration data for Cobbler's management resources: repos, files and packages. Mainly used by Koan to configure systems.

gen_config_data()
Generate configuration data for repos, files and packages.

Returns A dict which has all config data in it.

Return type dict

gen_config_data_for_koan()
Encode configuration data. Return json object for Koan.

Returns A json string for koan.

get_cobbler_resource(*resource*)
Wrapper around Cobbler blender method

Parameters **resource** – Not known what this actually is doing.

Returns Not known what this actually is doing.

resolve_resource_list(*list_data*)
Substitute variables in lists. Return new list.

Parameters **list_data** (*list*) – The list with the data to substitute.

Returns A list with the substituted data.

Return type list

resolve_resource_var(*string_data*)
Substitute variables in strings.

Parameters **string_data** – The string with the data to substitute.

Returns A str with the substituted data.

Return type str

8.11 cobbler.download_manager module

Cobbler DownloadManager

Copyright 2018, Jorgen Maas <jorgen.maas@gmail.com>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.download_manager.**DownloadManager**(*collection_mgr*, *logger=None*)
Bases: object

download_file(*url*, *dst*, *proxies=None*, *cert=None*)
Donwload a file from a URL and save it to any disc location.

Parameters

- **url** – The URL the request.
- **dst** – The destination file path.
- **proxies** – Override the default Cobbler proxies.
- **cert** – Override the default Cobbler certs.

urlread(*url*, *proxies=None*, *cert=None*)
Read the content of a given URL and pass the requests. Response object to the caller.

Parameters

- **url** – The URL the request.
- **proxies** – Override the default Cobbler proxies.
- **cert** – Override the default Cobbler certs.

Returns The Python `requests.Response` object.

8.12 cobbler.field_info module

Deprecated fields that have been renamed, but we need to account for them appearing in older datastructs that may not have been saved since the code change.

8.13 cobbler.module_loader module

Module loader, adapted for Cobbler usage

Copyright 2006-2009, Red Hat, Inc and Others Adrian Likins <alikins@redhat.com> Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.module_loader.**get_module_by_name**(*name*)
Get a module by its name. The category of the module is not needed.

Parameters **name** – The name of the module.

Returns The module asked by the function parameter.

cobbler.module_loader.**get_module_from_file**(*category*, *field*, *fallback_module_name=None*)
Get Python module, based on name defined in configuration file

Parameters

- **category** (*str*) – field category in configuration file
- **field** (*str*) – field in configuration file
- **fallback_module_name** (*str*) – default value used if category/field is not found in configuration file

Raises ***CX*** – If unable to load Python module

Returns A Python module.

cobbler.module_loader.**get_module_name**(*category*, *field*, *fallback_module_name=None*)
Get module name from configuration file

Parameters

- **category** (*str*) – Field category in configuration file.

- **field** (*str*) – Field in configuration file
- **fallback_module_name** (*str*) – Default value used if category/field is not found in configuration file

Raises *CX* – if unable to find configuration file

Returns module name

Return type str

cobbler.module_loader.**get_modules_in_category**(*category*)

Return all modules of a module category.

Parameters **category** – The category.

Returns A list of all modules of that category. Returns an empty list if the Category does not exist.

Return type list

cobbler.module_loader.**load_modules**(*module_path='/home/docs/checkouts/readthedocs.org/user_builds/cobbler/che*
blacklist=None)

Load the modules from the path handed to the function into Cobbler.

Parameters

- **module_path** – The path which should be considered as the root module path.
- **blacklist** – Currently an unused parameter.

Returns Two dictionary's with the dynamically loaded modules.

8.14 cobbler.power_manager module

Power management library. Encapsulate the logic to run power management commands so that the Cobbler user does not have to remember different power management tools syntaxes. This makes rebooting a system for OS installation much easier.

class cobbler.power_manager.**PowerManager**(*api*, *collection_mgr*, *logger=None*)

Bases: object

Handles power management in systems

get_power_status(*system*, *user=None*, *password=None*, *logger=None*)

Get power status for a system that has power management configured.

Parameters

- **system** (*System*) – Cobbler system
- **user** (*str*) – power management user
- **password** (*str*) – power management password
- **logger** (*Logger*) – logger

Returns if system is powered on

Return type bool

power_off(*system*, *user=None*, *password=None*, *logger=None*)
Powers down a system that has power management configured.

Parameters

- **system** (System) – Cobbler system
- **user** (*str*) – power management user
- **password** (*str*) – power management password
- **logger** (Logger) – logger

power_on(*system*, *user=None*, *password=None*, *logger=None*)
Powers up a system that has power management configured.

Parameters

- **system** (System) – Cobbler system
- **user** (*str*) – power management user
- **password** (*str*) – power management password
- **logger** (Logger) – logger

reboot(*system*, *user=None*, *password=None*, *logger=None*)
Reboot a system that has power management configured.

Parameters

- **system** (System) – Cobbler system
- **user** (*str*) – power management user
- **password** (*str*) – power management password
- **logger** (Logger) – logger

cobbler.power_manager.**get_power_command**(*power_type*)
Get power management command path

Parameters **power_type** (*str*) – power management type

Returns power management command path

Return type str or None

cobbler.power_manager.**get_power_types**()
Get possible power management types.

Returns Possible power management types

Return type list

cobbler.power_manager.**validate_power_type**(*power_type*)
Check if a power management type is valid.

Parameters **power_type** (*str*) – Power management type.

Raises ***CX*** – if power management type is invalid

8.15 cobbler.remote module

Copyright 2007-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.remote.**CobblerThread**(*event_id*, *remote*, *logatron*, *options*, *task_name*, *api*)
Bases: threading.Thread

Code for Cobbler's XMLRPC API.

on_done()
This stub is needed to satisfy the Python inheritance chain.

run()
Run the thread.

Returns The return code of the action. This may possibly a boolean or a Linux return code.

class cobbler.remote.**CobblerXMLRPCInterface**(*api*)
Bases: object

This is the interface used for all XMLRPC methods, for instance, as used by koan or CobblerWeb.

Most read-write operations require a token returned from "login". Read operations do not.

background_aclsetup(*options*, *token*)
Get the acl configuration from the config and set the acls in the backgroud.

Parameters

- **options** – Not known what this parameter does.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

background_buildiso(*options*, *token*)
Generates an ISO in /var/www/cobbler/pub that can be used to install profiles without using PXE.

Parameters

- **options** – Not known what this parameter does.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

background_dlcontent(*options*, *token*)
Download bootloaders and other support files.

Parameters

- **options** – Unknown what this parameter is doing at the moment.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

background_hardlink(*options*, *token*)
Hardlink all files as a background task.

Parameters

- **options** – Not known what this parameter does.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

background_import(*options*, *token*)
Import an ISO image in the background.

Parameters

- **options** – Not known what this parameter does.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

background_power_system(*options*, *token*)
Power a system asynchronously in the background.

Parameters

- **options** – Not known what this parameter does.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

background_replicate(*options*, *token*)
Replicate Cobbler in the background to another Cobbler instance.

Parameters

- **options** – Not known what this parameter does.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

background_reposync(*options*, *token*)
Run a reposync in the background.

Parameters

- **options** – Not known what this parameter does.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

background_signature_update(*options*, *token*)
Run a signature update in the background.

Parameters

- **options** – Not known what this parameter does.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

background_sync(*options*, *token*)
Run a full Cobbler sync in the background.

Parameters

- **options** – Not known what this parameter does.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

background_validate_autoinstall_files(*options*, *token*)
Validate all autoinstall files in the background.

Parameters

- **options** – Not known what this parameter does.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The id of the task which was started.

Return type str

check(*token*)
Returns a list of all the messages/warnings that are things that admin may want to correct about the configuration of the Cobbler server. This has nothing to do with "check_access" which is an auth/authz function in the XMLRPC API.

Parameters **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns None or a list of things to address.

Return type None or list

check_access(*token*, *resource*, *arg1=None*, *arg2=None*)
Check if the token which was provided has access.

Parameters

- **token** – The token to check access for.
- **resource** – The resource for which access shall be checked.
- **arg1** – Arguments to hand to the authorization provider.
- **arg2** – Arguments to hand to the authorization provider.

Returns Whether the authentication was successful or not.

check_access_no_fail(*token*, *resource*, *arg1=None*, *arg2=None*)
This is called by the WUI to decide whether an element is editable or not. It differs form check_access in that it is supposed to /not/ log the access checks (TBA) and does not raise exceptions.

Parameters

- **token** – The token to check access for.
- **resource** – The resource for which access shall be checked.

- **arg1** – Arguments to hand to the authorization provider.
- **arg2** – Arguments to hand to the authorization provider.

Returns True if the object is editable or False otherwise.

Return type bool

clear_system_logs(*object_id*, *token=None*, *logger=None*)
clears console logs of a system

Parameters

- **object_id** – The object id of the system to clear the logs of.
- **token** – The API-token obtained via the login() method.
- **logger** – The logger to audit all actions with.

Returns True if the operation succeeds.

copy_distro(*object_id*, *newname*, *token=None*)
Copies a distribution and renames it afterwards.

Parameters

- **object_id** – The object id of the item in question.
- **newname** – The new name for the copied object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

copy_file(*object_id*, *newname*, *token=None*)
Copies a file and rename it afterwards.

Parameters

- **object_id** – The object id of the item in question.
- **newname** – The new name for the copied object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

copy_image(*object_id*, *newname*, *token=None*)
Copies an image and renames it afterwards.

Parameters

- **object_id** – The object id of the item in question.
- **newname** – The new name for the copied object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

copy_item(*what*, *object_id*, *newname*, *token=None*)
Creates a new object that matches an existing object, as specified by an id.

Parameters

- **what** – The item type which should be copied.
- **object_id** – The object id of the item in question.
- **newname** – The new name for the copied object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

copy_mgmtclass(*object_id*, *newname*, *token=None*)
Copies a management class and rename it afterwards.

Parameters

- **object_id** – The object id of the item in question.
- **newname** – The new name for the copied object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

copy_package(*object_id*, *newname*, *token=None*)
Copies a package and rename it afterwards.

Parameters

- **object_id** – The object id of the item in question.
- **newname** – The new name for the copied object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

copy_profile(*object_id*, *newname*, *token=None*)
Copies a profile and renames it afterwards.

Parameters

- **object_id** – The object id of the item in question.
- **newname** – The new name for the copied object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

copy_repo(*object_id*, *newname*, *token=None*)
Copies a repository and renames it afterwards.

Parameters

- **object_id** – The object id of the item in question.
- **newname** – The new name for the copied object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

copy_system(*object_id*, *newname*, *token=None*)
Copies a system and renames it afterwards.

Parameters

- **object_id** – The object id of the item in question.
- **newname** – The new name for the copied object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

disable_netboot(*name*, *token=None*, ***rest*)
This is a feature used by the pxe_just_once support, see manpage. Sets system named "name" to no-longer PXE. Disabled by default as this requires public API access and is technically a read-write operation.

Parameters

- **name** – The name of the system to disable netboot for.
- **token** – The API-token obtained via the login() method.

- **rest** – This parameter is unused.

Returns A boolean indicated the success of the action.

extended_version(*token=None*, ***rest*)
Returns the full dictionary of version information. See api.py for documentation.

Parameters

- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns The extended version of Cobbler

find_distro(*criteria=None*, *expand=False*, *token=None*, ***rest*)
Find a distro matching certain criteria.

Parameters

- **criteria** – The criteria a distribution needs to match.
- **expand** – Not only get the names but also the complete object in form of a dict.
- **token** – The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns All distributions which have matched the criteria.

find_file(*criteria=None*, *expand=False*, *token=None*, ***rest*)
Find a file matching certain criteria.

Parameters

- **criteria** – The criteria a distribution needs to match.
- **expand** – Not only get the names but also the complete object in form of a dict.
- **token** – The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns All files which have matched the criteria.

find_image(*criteria=None*, *expand=False*, *token=None*, ***rest*)
Find an image matching certain criteria.

Parameters

- **criteria** – The criteria a distribution needs to match.
- **expand** – Not only get the names but also the complete object in form of a dict.
- **token** – The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns All images which have matched the criteria.

find_items(*what*, *criteria=None*, *sort_field=None*, *expand=True*)
Works like get_items but also accepts criteria as a dict to search on.

Example: `{ "name" : "*.example.org" }`

Wildcards work as described by 'pydoc fnmatch'.

Parameters

- **what** – The object type to find.
- **criteria** – The criteria an item needs to match.
- **sort_field** – The field to sort the results after.

- **expand** (*bool*) – Not only get the names but also the complete object in form of a dict.

Returns A list of dicts.

Return type list

find_items_paged(*what*, *criteria=None*, *sort_field=None*, *page=None*, *items_per_page=None*, *token=None*)

Returns a list of dicts as with find_items but additionally supports returning just a portion of the total list, for instance in supporting a web app that wants to show a limited amount of items per page.

Parameters

- **what** – The object type to find.
- **criteria** – The criteria a distribution needs to match.
- **sort_field** – The field to sort the results after.
- **page** – The page to return
- **items_per_page** – The number of items per page.
- **token** – The API-token obtained via the login() method.

Returns The found items.

find_mgmtclass(*criteria=None*, *expand=False*, *token=None*, ***rest*)

Find a management class matching certain criteria.

Parameters

- **criteria** – The criteria a distribution needs to match.
- **expand** – Not only get the names but also the complete object in form of a dict.
- **token** – The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns All management classes which have matched the criteria.

find_package(*criteria=None*, *expand=False*, *token=None*, ***rest*)

Find a package matching certain criteria.

Parameters

- **criteria** – The criteria a distribution needs to match.
- **expand** – Not only get the names but also the complete object in form of a dict.
- **token** – The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns All packages which have matched the criteria.

find_profile(*criteria=None*, *expand=False*, *token=None*, ***rest*)

Find a profile matching certain criteria.

Parameters

- **criteria** – The criteria a distribution needs to match.
- **expand** – Not only get the names but also the complete object in form of a dict.
- **token** – The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns All profiles which have matched the criteria.

find_repo(*criteria=None*, *expand=False*, *token=None*, ***rest*)

Find a repository matching certain criteria.

Parameters

- **criteria** – The criteria a distribution needs to match.
- **expand** – Not only get the names but also the complete object in form of a dict.
- **token** – The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns All repositories which have matched the criteria.

find_system(*criteria=None*, *expand=False*, *token=None*, ***rest*)
Find a system matching certain criteria.

Parameters

- **criteria** – The criteria a distribution needs to match.
- **expand** – Not only get the names but also the complete object in form of a dict.
- **token** – The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns All systems which have matched the criteria.

find_system_by_dns_name(*dns_name*)
This is used by the puppet external nodes feature.

Parameters **dns_name** – The dns name of the system. This should be the fqdn and not only the hostname.

Returns All system information or an empty dict.

generate_autoinstall(*profile=None*, *system=None*, *REMOTE_ADDR=None*, *REMOTE_MAC=None*, ***rest*)
Generate the autoinstallation file and return it.

Parameters

- **profile** – The profile to generate the file for.
- **system** – The system to generate the file for.
- **REMOTE_ADDR** – This is dropped in this method since it is not needed here.
- **REMOTE_MAC** – This is dropped in this method since it is not needed here.
- **rest** – This is dropped in this method since it is not needed here.

Returns The str representation of the file.

generate_bootcfg(*profile=None*, *system=None*, ***rest*)
This generates the bootcfg for a system which is related to a certain profile.

Parameters

- **profile** – The profile which is associated to the system.
- **system** – The system which the bootcfg should be generated for.
- **rest** – This is dropped in this method since it is not needed here.

Returns The generated bootcfg.

generate_gpxe(*profile=None*, *system=None*, ***rest*)
Generate the gpx configuration.

Note: gPXE is deprecated and it is recommended to change to iPXE.

Parameters

- **profile** – The profile to generate gPXE config for.

- **system** – The system to generate gPXE config for.
- **rest** – This is dropped in this method since it is not needed here.

Returns The configuration as a str representation.

generate_profile_autoinstall (*profile*)
Generate a profile autoinstallation.

Parameters **profile** – The profile to generate the file for.

Returns The str representation of the file.

generate_script (*profile=None*, *system=None*, *name=None*, ***rest*)
Not known what this does exactly.

Parameters

- **profile** – Not known for what the profile is needed.
- **system** – Not known for what the system is needed.
- **name** – Name of the generated script.
- **rest** – This is dropped in this method since it is not needed here.

Returns Some generated script.

generate_system_autoinstall (*system*)
Generate a system autoinstallation.

Parameters **system** – The system to generate the file for.

Returns The str representation of the file.

get_authn_module_name (*token*)
Get the name of the currently used authentication module.

Parameters **token** – The API-token obtained via the login() method. Cobbler token, obtained form login()

Returns The name of the module.

get_autoinstall_snippets (*token=None*, ***rest*)
Returns all the automatic OS installation templates' snippets.

Parameters

- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns A list with all snippets.

get_autoinstall_templates (*token=None*, ***rest*)
Returns all of the automatic OS installation templates that are in use by the system.

Parameters

- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns A list with all templates.

get_blended_data (*profile=None*, *system=None*)
Combine all data which is available from a profile and system together and return it.

Parameters

- **profile** – The profile of the system.
- **system** – The system for which the data should be rendered.

Returns All values which could be blended together through the inheritance chain.

get_config_data(*hostname*)
Generate configuration data for the system specified by hostname.

Parameters **hostname** – The hostname for what to get the config data of.

Returns The config data as a json for Koan.

Return type str

get_distro(*name*, *flatten=False*, *token=None*, ***rest*)
Get a distribution.

Parameters

- **name** – The name of the distribution to get.
- **flatten** – If the item should be flattened.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – Not used with this method currently.

Returns The item or None.

get_distro_as_rendered(*name*, *token=None*, ***rest*)
Get distribution after passing through Cobbler's inheritance engine.

Parameters

- **name** (*str*) – distro name
- **token** (*str*) – authentication token
- **rest** – This is dropped in this method since it is not needed here.

Returns Get a template rendered as a distribution.

get_distro_for_koan(*name*, *token=None*, ***rest*)
This is a legacy function for 2.6.6 releases. :param name: The name of the distro to get. :param token: Auth token to authenticate against the api. :param rest: This is dropped in this method since it is not needed here. :return: The desired distro or '~'.

get_distro_handle(*name*, *token*)
Get a handle for a distribution which allows you to use the functions `modify_*` or `save_*` to manipulate it.

Parameters

- **name** – The name of the item.
- **token** – The API-token obtained via the login() method.

Returns The handle of the desired object.

get_distros(*page=None*, *results_per_page=None*, *token=None*, ***rest*)
This returns all distributions.

Parameters

- **page** – This parameter is not used currently.
- **results_per_page** – This parameter is not used currently.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns The list with all distros.

get_distros_since(*mtime*)
Return all of the distro objects that have been modified after mtime.

Parameters **mtime** – The time after which all items should be included. Everything before this will be excluded.

Returns The list of items which were modified after `mtime`.

get_event_log(*event_id*)
Returns the contents of a task log. Events that are not task-based do not have logs.

Parameters **event_id** – The event-id generated by Cobbler.

Returns The event log or a `?`.

Return type str

get_events(*for_user=''*)
Returns a dict(key=event id) = [statetime, name, state, [read_by_who]]

Parameters **for_user** (*str*) – (Optional) Filter events the user has not seen yet. If left unset, it will return all events.

Returns A dictionary with all the events (or all filtered events).

Return type dict

get_file(*name*, *flatten=False*, *token=None*, ***rest*)
Get a file.

Parameters

- **name** – The name of the file to get.
- **flatten** – If the item should be flattened.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – Not used with this method currently.

Returns The item or None.

get_file_as_rendered(*name*, *token=None*, ***rest*)
Get file after passing through Cobbler's inheritance engine

Parameters

- **name** (*str*) – file name
- **token** (*str*) – authentication token
- **rest** – This is dropped in this method since it is not needed here.

Returns Get a template rendered as a file.

get_file_for_koan(*name*, *token=None*, ***rest*)
This is a legacy function for 2.6.6 releases. :param name: Name of the file to get. :param token: Auth token to authenticate against the api. :param rest: This is dropped in this method since it is not needed here. :return: The desired file or '~'.

get_file_handle(*name*, *token*)
Get a handle for a file which allows you to use the functions `modify_*` or `save_*` to manipulate it.

Parameters

- **name** – The name of the item.
- **token** – The API-token obtained via the login() method.

Returns The handle of the desired object.

get_files(*page=None*, *results_per_page=None*, *token=None*, ***rest*)
This returns all files.

Parameters

- **page** – This parameter is not used currently.
- **results_per_page** – This parameter is not used currently.
- **token** – The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns The list of all files.

get_files_since(*mtime*)
See documentation for get_distros_since

Parameters **mtime** – The time after which all items should be included. Everything before this will be excluded.

Returns The list of items which were modified after `mtime`.

get_image(*name*, *flatten=False*, *token=None*, ***rest*)
Get an image.

Parameters

- **name** – The name of the image to get.
- **flatten** – If the item should be flattened.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – Not used with this method currently.

Returns The item or None.

get_image_as_rendered(*name*, *token=None*, ***rest*)
Get repository after passing through Cobbler's inheritance engine.

Parameters

- **name** (*str*) – image name
- **token** (*str*) – authentication token
- **rest** – This is dropped in this method since it is not needed here.

Returns Get a template rendered as an image.

get_image_for_koan(*name*, *token=None*, ***rest*)
This is a legacy function for 2.6.6 releases. :param name: The name of the image to get. :param token: Auth token to authenticate against the api. :param rest: This is dropped in this method since it is not needed here. :return: The desired image or '~'

get_image_handle(*name*, *token*)
Get a handle for an image which allows you to use the functions `modify_*` or `save_*` to manipulate it.

Parameters

- **name** – The name of the item.
- **token** – The API-token obtained via the login() method.

Returns The handle of the desired object.

get_images(*page=None*, *results_per_page=None*, *token=None*, ***rest*)
This returns all images.

Parameters

- **page** – This parameter is not used currently.
- **results_per_page** – This parameter is not used currently.

- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns The list of all images.

get_images_since(*mtime*)
See documentation for get_distros_since

Parameters **mtime** – The time after which all items should be included. Everything before this will be excluded.

Returns The list of items which were modified after `mtime`.

get_item(*what*, *name*, *flatten=False*)
Returns a dict describing a given object.

Parameters

- **what** – "distro", "profile", "system", "image", "repo", etc
- **name** – the object name to retrieve
- **flatten** – reduce dicts to string representations (True/False)

Returns The item or None.

get_item_handle(*what*, *name*, *token=None*)
Given the name of an object (or other search parameters), return a reference (object id) that can be used with `modify_*` functions or `save_*` functions to manipulate that object.

Parameters

- **what** – The collection where the item is living in.
- **name** – The name of the item.
- **token** – The API-token obtained via the login() method.

Returns The handle of the desired object.

get_item_names(*what*)
This is just like get_items, but transmits less data.

Parameters **what** – is the name of a Cobbler object type, as described for get_item.

Returns Returns a list of object names (keys) for the given object type.

get_items(*what*)
Individual list elements are the same for get_item.

Parameters **what** – is the name of a Cobbler object type, as described for get_item.

Returns This returns a list of dicts.

get_mgmtclass(*name*, *flatten=False*, *token=None*, ***rest*)
Get a management class.

Parameters

- **name** – The name of the management class to get.
- **flatten** – If the item should be flattened.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – Not used with this method currently.

Returns The item or None.

get_mgmtclass_as_rendered(*name*, *token=None*, ***rest*)
Get management class after passing through Cobbler's inheritance engine

Parameters

- **name** (*str*) – management class name
- **token** (*str*) – authentication token
- **rest** – This is dropped in this method since it is not needed here.

Returns Get a template rendered as a management class.

get_mgmtclass_for_koan(*name*, *token=None*, ***rest*)
This is a legacy function for 2.6.6 releases. :param name: Name of the mgmtclass to get. :param token: Auth token to authenticate against the api. :param rest: This is dropped in this method since it is not needed here. :return: The desired mgmtclass or ~.

get_mgmtclass_handle(*name*, *token*)
Get a handle for a management class which allows you to use the functions `modify_*` or `save_*` to manipulate it.

Parameters

- **name** – The name of the item.
- **token** – The API-token obtained via the login() method.

Returns The handle of the desired object.

get_mgmtclasses(*page=None*, *results_per_page=None*, *token=None*, ***rest*)
This returns all managementclasses.

Parameters

- **page** – This parameter is not used currently.
- **results_per_page** – This parameter is not used currently.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns The list of all managementclasses.

get_mgmtclasses_since(*mtime*)
See documentation for get_distros_since

Parameters **mtime** – The time after which all items should be included. Everything before this will be excluded.

Returns The list of items which were modified after `mtime`.

get_package(*name*, *flatten=False*, *token=None*, ***rest*)
Get a package.

Parameters

- **name** – The name of the package to get.
- **flatten** – If the item should be flattened.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – Not used with this method currently.

Returns The item or None.

get_package_as_rendered(*name*, *token=None*, ***rest*)
Get package after passing through Cobbler's inheritance engine

Parameters

- **name** (*str*) – package name

- **token** (*str*) – authentication token
- **rest** – This is dropped in this method since it is not needed here.

Returns Get a template rendered as a package.

get_package_for_koan(*name*, *token=None*, ***rest*)
This is a legacy function for 2.6.6 releases. :param name: Name of the package to get. :param token: Auth token to authenticate against the api. :param rest: This is dropped in this method since it is not needed here. :return: The desired package or '~'.

get_package_handle(*name*, *token*)
Get a handle for a package which allows you to use the functions `modify_*` or `save_*` to manipulate it.

Parameters

- **name** – The name of the item.
- **token** – The API-token obtained via the login() method.

Returns The handle of the desired object.

get_packages(*page=None*, *results_per_page=None*, *token=None*, ***rest*)
This returns all packages.

Parameters

- **page** – This parameter is not used currently.
- **results_per_page** – This parameter is not used currently.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns The list of all packages tracked in Cobbler.

get_packages_since(*mtime*)
See documentation for get_distros_since

Parameters **mtime** – The time after which all items should be included. Everything before this will be excluded.

Returns The list of items which were modified after `mtime`.

get_profile(*name*, *flatten=False*, *token=None*, ***rest*)
Get a profile.

Parameters

- **name** – The name of the profile to get.
- **flatten** – If the item should be flattened.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – Not used with this method currently.

Returns The item or None.

get_profile_as_rendered(*name*, *token=None*, ***rest*)
Get profile after passing through Cobbler's inheritance engine.

Parameters

- **name** (*str*) – profile name
- **token** (*str*) – authentication token
- **rest** – This is dropped in this method since it is not needed here.

Returns Get a template rendered as a profile.

get_profile_for_koan(*name*, *token=None*, ***rest*)
This is a legacy function for 2.6.6 releases. :param name: The name of the profile to get. :param token: Auth token to authenticate against the api. :param rest: This is dropped in this method since it is not needed here. :return: The desired profile or '~'.

get_profile_handle(*name*, *token*)
Get a handle for a profile which allows you to use the functions `modify_*` or `save_*` to manipulate it.

Parameters

- **name** – The name of the item.
- **token** – The API-token obtained via the login() method.

Returns The handle of the desired object.

get_profiles(*page=None*, *results_per_page=None*, *token=None*, ***rest*)
This returns all profiles.

Parameters

- **page** – This parameter is not used currently.
- **results_per_page** – This parameter is not used currently.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns The list with all profiles.

get_profiles_since(*mtime*)
See documentation for get_distros_since

Parameters **mtime** – The time after which all items should be included. Everything before this will be excluded.

Returns The list of items which were modified after `mtime`.

get_random_mac(*virt_type='xenpv'*, *token=None*, ***rest*)
Wrapper for `utils.get_random_mac()`. Used in the webui.

Parameters

- **virt_type** – The type of the virtual machine.
- **token** – The API-token obtained via the login() method. Auth token to authenticate against the api.
- **rest** – This is dropped in this method since it is not needed here.

Returns The random mac address which shall be used somewhere else.

get_repo(*name*, *flatten=False*, *token=None*, ***rest*)
Get a repository.

Parameters

- **name** – The name of the repository to get.
- **flatten** – If the item should be flattened.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – Not used with this method currently.

Returns The item or None.

get_repo_as_rendered(*name*, *token=None*, ***rest*)
Get repository after passing through Cobbler's inheritance engine.

Parameters

- **name** (*str*) – repository name
- **token** (*str*) – authentication token
- **rest** – This is dropped in this method since it is not needed here.

Returns Get a template rendered as a repository.

get_repo_config_for_profile(*profile_name*, ***rest*)
Return the yum configuration a given profile should use to obtain all of it's Cobbler associated repos.

Parameters

- **profile_name** – The name of the profile for which the repository config should be looked up.
- **rest** – This is dropped in this method since it is not needed here.

Returns The repository configuration for the profile.

get_repo_config_for_system(*system_name*, ***rest*)
Return the yum configuration a given profile should use to obtain all of it's Cobbler associated repos.

Parameters

- **system_name** – The name of the system for which the repository config should be looked up.
- **rest** – This is dropped in this method since it is not needed here.

Returns The repository configuration for the system.

get_repo_for_koan(*name*, *token=None*, ***rest*)
This is a legacy function for 2.6.6 releases. :param name: The name of the repo to get. :param token: Auth token to authenticate against the api. :param rest: This is dropped in this method since it is not needed here. :return: The desired repo or '~'.

get_repo_handle(*name*, *token*)
Get a handle for a repository which allows you to use the functions `modify_*` or `save_*` to manipulate it.

Parameters

- **name** – The name of the item.
- **token** – The API-token obtained via the login() method.

Returns The handle of the desired object.

get_repos(*page=None*, *results_per_page=None*, *token=None*, ***rest*)
This returns all repositories.

Parameters

- **page** – This parameter is not used currently.
- **results_per_page** – This parameter is not used currently.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns The list of all repositories.

get_repos_compatible_with_profile(*profile=None*, *token=None*, ***rest*)
Get repos that can be used with a given profile name.

Parameters

- **profile** – The profile to check for compatibility.
- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns The list of compatible repositories.

Return type list

get_repos_since(*mtime*)
See documentation for get_distros_since

Parameters **mtime** – The time after which all items should be included. Everything before this will be excluded.

Returns The list of items which were modified after `mtime`.

get_settings(*token=None*, ***rest*)
Return the contents of /etc/cobbler/settings, which is a dict.

Parameters

- **token** – The API-token obtained via the login() method.
- **rest** – Unused parameter.

Returns Get the settings which are currently in Cobbler present.

get_signatures(*token=None*, ***rest*)
Return the contents of the API signatures

Parameters

- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns Get the content of the currently loaded signatures file.

get_status(*mode='normal'*, *token=None*, ***rest*)
Returns the same information as *cobbler status* While a read-only operation, this requires a token because it's potentially a fair amount of I/O

Parameters

- **mode** – How the status should be presented.
- **token** – The API-token obtained via the login() method. Auth token to authenticate against the api.
- **rest** – This parameter is currently unused for this method.

Returns The human or machine readable status of the status of Cobbler.

get_system(*name*, *flatten=False*, *token=None*, ***rest*)
Get a system.

Parameters

- **name** – The name of the system to get.
- **flatten** – If the item should be flattened.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – Not used with this method currently.

Returns The item or None.

get_system_as_rendered(*name*, *token=None*, ***rest*)
Get profile after passing through Cobbler's inheritance engine.

Parameters

- **name** (*str*) – system name
- **token** (*str*) – authentication token
- **rest** – This is dropped in this method since it is not needed here.

Returns Get a template rendered as a system.

get_system_for_koan(*name*, *token=None*, ***rest*)
This is a legacy function for 2.6.6 releases. :param name: The name of the system to get. :param token: Auth token to authenticate against the api. :param rest: This is dropped in this method since it is not needed here. :return: The desired system or '~'.

get_system_handle(*name*, *token*)
Get a handle for a system which allows you to use the functions `modify_*` or `save_*` to manipulate it.

Parameters

- **name** – The name of the item.
- **token** – The API-token obtained via the login() method.

Returns The handle of the desired object.

get_systems(*page=None*, *results_per_page=None*, *token=None*, ***rest*)
This returns all Systems.

Parameters

- **page** – This parameter is not used currently.
- **results_per_page** – This parameter is not used currently.
- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.
- **rest** – This parameter is not used currently.

Returns The list of all systems.

get_systems_since(*mtime*)
See documentation for get_distros_since

Parameters **mtime** – The time after which all items should be included. Everything before this will be excluded.

Returns The list of items which were modified after `mtime`.

get_task_status(*event_id*)
Get the current status of the task.

Parameters **event_id** – The unique id of the task.

Returns The event status.

get_template_file_for_profile(*profile_name*, *path*, ***rest*)
Return the templated file requested for this profile

Parameters

- **profile_name** – The name of the profile to get the template file for.
- **path** – The path to the template which is requested.
- **rest** – This is dropped in this method since it is not needed here.

Returns The template file as a str representation.

get_template_file_for_system(*system_name*, *path*, ***rest*)
Return the templated file requested for this system

Parameters

- **system_name** – The name of the system to get the template file for.
- **path** – The path to the template which is requested.
- **rest** – This is dropped in this method since it is not needed here.

Returns The template file as a str representation.

get_user_from_token(*token*)
Given a token returned from login, return the username that logged in with it.

Parameters **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns The username if the token was valid.

Raises ***CX*** – If the token supplied to the function is invalid.

get_valid_archs(*token=None*)
Return the list of valid architectures as read in from the distro signatures data

Parameters **token** – The API-token obtained via the login() method.

Returns Get a list of all valid architectures.

get_valid_breeds(*token=None*, ***rest*)
Return the list of valid breeds as read in from the distro signatures data

Parameters

- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns All valid OS-Breeds which are present in Cobbler.

get_valid_os_versions(*token=None*, ***rest*)
Return the list of valid os_versions as read in from the distro signatures data

Parameters

- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns Get all valid OS-Versions

get_valid_os_versions_for_breed(*breed*, *token=None*, ***rest*)
Return the list of valid os_versions for the given breed

Parameters

- **breed** – The OS-Breed which is requested.
- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns All valid OS-versions for a certain breed.

has_item(*what*, *name*, *token=None*)
Returns True if a given collection has an item with a given name, otherwise returns False.

Parameters

- **what** – The collection to search through.
- **name** – The name of the item.
- **token** – The API-token obtained via the login() method.

Returns True if item was found, otherwise False.

is_autoinstall_in_use(*ai*, *token=None*, ***rest*)
Check if the autoinstall for a system is in use.

Parameters

- **ai** – The name of the system which could potentially be in autoinstall mode.
- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns True if this is the case, otherwise False.

last_modified_time(*token=None*)
Return the time of the last modification to any object. Used to verify from a calling application that no Cobbler objects have changed since last check. This method is implemented in the module api under the same name.

Parameters **token** – The API-token obtained via the login() method. The API-token obtained via the login() method.

Returns 0 if there is no file where the information required for this method is saved.

Return type float

login(*login_user*, *login_password*)
Takes a username and password, validates it, and if successful returns a random login token which must be used on subsequent method calls. The token will time out after a set interval if not used. Re-logging in permitted.

Parameters

- **login_user** – The username which is used to authenticate at Cobbler.
- **login_password** – The password which is used to authenticate at Cobbler.

Returns The token which can be used further on.

logout(*token*)
Retires a token ahead of the timeout.

Parameters **token** – The API-token obtained via the login() method. Cobbler token, obtained form login()

Returns if operation was successful or not

Return type bool

modify_distro(*object_id*, *attribute*, *arg*, *token*)
Modify a single attribute of a distribution.

Parameters

- **object_id** – The id of the object which shall be modified.
- **attribute** – The attribute name which shall be edited.
- **arg** – The new value for the arguement.
- **token** – The API-token obtained via the login() method.

Returns True if the action was successful. Otherwise False.

modify_file(*object_id*, *attribute*, *arg*, *token*)
Modify a single attribute of a file.

Parameters

- **object_id** – The id of the object which shall be modified.
- **attribute** – The attribute name which shall be edited.

- **arg** – The new value for the arguement.
- **token** – The API-token obtained via the login() method.

Returns True if the action was successful. Otherwise False.

modify_image(*object_id*, *attribute*, *arg*, *token*)
Modify a single attribute of an image.

Parameters

- **object_id** – The id of the object which shall be modified.
- **attribute** – The attribute name which shall be edited.
- **arg** – The new value for the arguement.
- **token** – The API-token obtained via the login() method.

Returns True if the action was successful. Otherwise False.

modify_item(*what*, *object_id*, *attribute*, *arg*, *token*)
Adjusts the value of a given field, specified by 'what' on a given object id. Allows modification of certain attributes on newly created or existing distro object handle.

Parameters

- **what** – The type of object to modify.1
- **object_id** – The id of the object which shall be modified.
- **attribute** – The attribute name which shall be edited.
- **arg** – The new value for the arguement.
- **token** – The API-token obtained via the login() method.

Returns True if the action was successful. Otherwise False.

modify_mgmtclass(*object_id*, *attribute*, *arg*, *token*)
Modify a single attribute of a managementclass.

Parameters

- **object_id** – The id of the object which shall be modified.
- **attribute** – The attribute name which shall be edited.
- **arg** – The new value for the arguement.
- **token** – The API-token obtained via the login() method.

Returns True if the action was successful. Otherwise False.

modify_package(*object_id*, *attribute*, *arg*, *token*)
Modify a single attribute of a package.

Parameters

- **object_id** – The id of the object which shall be modified.
- **attribute** – The attribute name which shall be edited.
- **arg** – The new value for the arguement.
- **token** – The API-token obtained via the login() method.

Returns True if the action was successful. Otherwise False.

modify_profile(*object_id*, *attribute*, *arg*, *token*)
Modify a single attribute of a profile.

Parameters

- **object_id** – The id of the object which shall be modified.

- **attribute** – The attribute name which shall be edited.
- **arg** – The new value for the arguement.
- **token** – The API-token obtained via the login() method.

Returns True if the action was successful. Otherwise False.

modify_repo(*object_id*, *attribute*, *arg*, *token*)
Modify a single attribute of a repository.

Parameters

- **object_id** – The id of the object which shall be modified.
- **attribute** – The attribute name which shall be edited.
- **arg** – The new value for the arguement.
- **token** – The API-token obtained via the login() method.

Returns True if the action was successful. Otherwise False.

modify_setting(*setting_name*, *value*, *token*)
Modify a single attribute of a setting.

Parameters

- **setting_name** – The name of the setting which shall be adjusted.
- **value** – The new value for the setting.
- **token** – The API-token obtained via the login() method.

Returns 0 on success, 1 on error.

modify_system(*object_id*, *attribute*, *arg*, *token*)
Modify a single attribute of a system.

Parameters

- **object_id** – The id of the object which shall be modified.
- **attribute** – The attribute name which shall be edited.
- **arg** – The new value for the arguement.
- **token** – The API-token obtained via the login() method.

Returns True if the action was successful. Otherwise False.

new_distro(*token*)
See `new_item()`.

Parameters **token** – The API-token obtained via the login() method.

Returns The object id for the newly created object.

new_file(*token*)
See `new_item()`.

Parameters **token** – The API-token obtained via the login() method.

Returns The object id for the newly created object.

new_image(*token*)
See `new_item()`.

Parameters **token** – The API-token obtained via the login() method.

Returns The object id for the newly created object.

new_item(*what*, *token*, *is_subobject=False*)
Creates a new (unconfigured) object, returning an object handle that can be used.

Creates a new (unconfigured) object, returning an object handle that can be used with `modify_*` methods and then finally `save_*` methods. The handle only exists in memory until saved.

Parameters

- **what** – specifies the type of object: `distro`, `profile`, `system`, `repo`, or `image`
- **token** – The API-token obtained via the login() method.
- **is_subobject** – If the object is a subobject of an already existing object or not.

Returns The object id for the newly created object.

new_mgmtclass(*token*)
See `new_item()`.

Parameters **token** – The API-token obtained via the login() method.

Returns The object id for the newly created object.

new_package(*token*)
See `new_item()`.

Parameters **token** – The API-token obtained via the login() method.

Returns The object id for the newly created object.

new_profile(*token*)
See `new_item()`.

Parameters **token** – The API-token obtained via the login() method.

Returns The object id for the newly created object.

new_repo(*token*)
See `new_item()`.

Parameters **token** – The API-token obtained via the login() method.

Returns The object id for the newly created object.

new_subprofile(*token*)
See `new_item()`.

Parameters **token** – The API-token obtained via the login() method.

Returns The object id for the newly created object.

new_system(*token*)
See `new_item()`.

Parameters **token** – The API-token obtained via the login() method.

Returns The object id for the newly created object.

ping()
Deprecated method. Now does nothing.

Returns Always True

Return type bool

power_system(*system_id*, *power*, *token*)
Execute power task synchronously.

Returns true if the operation succeeded or if the system is powered on (in case of status). False otherwise.

Parameters

- **token** – The API-token obtained via the login() method. The API-token obtained via the login() method. All tasks require tokens.
- **system_id** – system handle
- **power** – power operation (on/off/status/reboot)

Return type bool

read_autoinstall_snippet(*file_path*, *token*)
Read an automatic OS installation snippet file

Parameters

- **file_path** (*str*) – automatic OS installation snippet file path
- **token** – The API-token obtained via the login() method. Cobbler token, obtained form login()

Returns file content

Return type str

read_autoinstall_template(*file_path*, *token*)
Read an automatic OS installation template file

Parameters

- **file_path** (*str*) – automatic OS installation template file path
- **token** – The API-token obtained via the login() method. Cobbler token, obtained form login()

Returns file content

Return type str

register_new_system(*info*, *token=None*, ***rest*)
If register_new_installs is enabled in settings, this allows /usr/bin/cobbler-register (part of the koan package) to add new system records remotely if they don't already exist. There is a cobbler_register snippet that helps with doing this automatically for new installs but it can also be used for existing installs.

See "AutoRegistration" on the Wiki.

Parameters

- **info** – The system information which is provided by the system.
- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns Return 0 if everything succeeded.

remove_autoinstall_snippet(*file_path*, *token*)
Remove an automated OS installation snippet file

Parameters

- **file_path** (*str*) – automated OS installation snippet file path
- **token** – Cobbler token, obtained form login()

Returns bool if operation was successful

remove_autoinstall_template(*file_path*, *token*)
Remove an automatic OS installation template file

Parameters

- **file_path** (*str*) – automatic OS installation template file path

- **token** – The API-token obtained via the login() method. Cobbler token, obtained form login()

Returns bool if operation was successful

remove_distro(*name*, *token*, *recursive=True*)
Deletes a distribution from Cobbler.

Parameters

- **name** – The name of the item to remove.
- **token** – The API-token obtained via the login() method.
- **recursive** (*bool*) – If items which are depending on this one should be erased too.

Returns True if the action was successful.

remove_file(*name*, *token*, *recursive=True*)
Deletes a file from Cobbler.

Parameters

- **name** – The name of the item to remove.
- **token** – The API-token obtained via the login() method.
- **recursive** (*bool*) – If items which are depending on this one should be erased too.

Returns True if the action was successful.

remove_image(*name*, *token*, *recursive=True*)
Deletes an image from Cobbler.

Parameters

- **name** – The name of the item to remove.
- **token** – The API-token obtained via the login() method.
- **recursive** (*bool*) – If items which are depending on this one should be erased too.

Returns True if the action was successful.

remove_item(*what*, *name*, *token*, *recursive=True*)
Deletes an item from a collection. Note that this requires the name of the distro, not an item handle.

Parameters

- **what** – The item type of the item to remove.
- **name** – The name of the item to remove.
- **token** – The API-token obtained via the login() method.
- **recursive** (*bool*) – If items which are depending on this one should be erased too.

Returns True if the action was successful.

remove_mgmtclass(*name*, *token*, *recursive=True*)
Deletes a managementclass from Cobbler.

Parameters

- **name** – The name of the item to remove.
- **token** – The API-token obtained via the login() method.
- **recursive** (*bool*) – If items which are depending on this one should be erased too.

Returns True if the action was successful.

remove_package(*name*, *token*, *recursive=True*)
Deletes a package from Cobbler.

Parameters

- **name** – The name of the item to remove.
- **token** – The API-token obtained via the login() method.
- **recursive** (*bool*) – If items which are depending on this one should be erased too.

Returns True if the action was successful.

remove_profile(*name*, *token*, *recursive=True*)
Deletes a profile from Cobbler.

Parameters

- **name** – The name of the item to remove.
- **token** – The API-token obtained via the login() method.
- **recursive** (*bool*) – If items which are depending on this one should be erased too.

Returns True if the action was successful.

remove_repo(*name*, *token*, *recursive=True*)
Deletes a repository from Cobbler.

Parameters

- **name** – The name of the item to remove.
- **token** – The API-token obtained via the login() method.
- **recursive** (*bool*) – If items which are depending on this one should be erased too.

Returns True if the action was successful.

remove_system(*name*, *token*, *recursive=True*)
Deletes a system from Cobbler.

Parameters

- **name** – The name of the item to remove.
- **token** – The API-token obtained via the login() method.
- **recursive** (*bool*) – If items which are depending on this one should be erased too.

Returns True if the action was successful.

rename_distro(*object_id*, *newname*, *token=None*)
Renames a distribution specified by object_id to a new name.

Parameters

- **object_id** – The id which refers to the object.
- **newname** – The new name for the object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

rename_file(*object_id*, *newname*, *token=None*)
Renames a file specified by object_id to a new name.

Parameters

- **object_id** – The id which refers to the object.
- **newname** – The new name for the object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

rename_image(*object_id*, *newname*, *token=None*)
Renames an image specified by object_id to a new name.

Parameters

- **object_id** – The id which refers to the object.
- **newname** – The new name for the object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

rename_item(*what*, *object_id*, *newname*, *token=None*)
Renames an object specified by object_id to a new name.

Parameters

- **what** – The type of object which shall be renamed to a new name.
- **object_id** – The id which refers to the object.
- **newname** – The new name for the object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

rename_mgmtclass(*object_id*, *newname*, *token=None*)
Renames a managementclass specified by object_id to a new name.

Parameters

- **object_id** – The id which refers to the object.
- **newname** – The new name for the object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

rename_package(*object_id*, *newname*, *token=None*)
Renames a package specified by object_id to a new name.

Parameters

- **object_id** – The id which refers to the object.
- **newname** – The new name for the object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

rename_profile(*object_id*, *newname*, *token=None*)
Renames a profile specified by object_id to a new name.

Parameters

- **object_id** – The id which refers to the object.
- **newname** – The new name for the object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

rename_repo(*object_id*, *newname*, *token=None*)
Renames a repository specified by object_id to a new name.

Parameters

- **object_id** – The id which refers to the object.
- **newname** – The new name for the object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

rename_system(*object_id*, *newname*, *token=None*)
Renames a system specified by object_id to a new name.

Parameters

- **object_id** – The id which refers to the object.
- **newname** – The new name for the object.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

run_install_triggers(*mode*, *objtype*, *name*, *ip*, *token=None*, ***rest*)
This is a feature used to run the pre/post install triggers. See CobblerTriggers on Wiki for details

Parameters

- **mode** – The mode of the triggers. May be "pre", "post" or "firstboot".
- **objtype** – The type of object. This should correspond to the collection type.
- **name** – The name of the object.
- **ip** – The ip of the objet.
- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns True if everything worked correctly.

save_distro(*object_id*, *token*, *editmode='bypass'*)
Saves a newly created or modified object to disk. Calling save is required for any changes to persist.

Parameters

- **object_id** – The id of the object to save.
- **token** – The API-token obtained via the login() method.
- **editmode** – The mode which shall be used to persist the changes. Currently "new" and "bypass" are supported.

Returns True if the action succeeded.

save_file(*object_id*, *token*, *editmode='bypass'*)
Saves a newly created or modified object to disk. Calling save is required for any changes to persist.

Parameters

- **object_id** – The id of the object to save.
- **token** – The API-token obtained via the login() method.
- **editmode** – The mode which shall be used to persist the changes. Currently "new" and "bypass" are supported.

Returns True if the action succeeded.

save_image(*object_id*, *token*, *editmode='bypass'*)
Saves a newly created or modified object to disk. Calling save is required for any changes to persist.

Parameters

- **object_id** – The id of the object to save.
- **token** – The API-token obtained via the login() method.
- **editmode** – The mode which shall be used to persist the changes. Currently "new" and "bypass" are supported.

Returns True if the action succeeded.

save_item(*what*, *object_id*, *token*, *editmode='bypass'*)
Saves a newly created or modified object to disk. Calling save is required for any changes to persist.

Parameters

- **what** – The type of object which shall be saved. This corresponds to the collections.
- **object_id** – The id of the object to save.
- **token** – The API-token obtained via the login() method.
- **editmode** – The mode which shall be used to persist the changes. Currently "new" and "bypass" are supported.

Returns True if the action succeeded.

save_mgmtclass(*object_id*, *token*, *editmode='bypass'*)
Saves a newly created or modified object to disk. Calling save is required for any changes to persist.

Parameters

- **object_id** – The id of the object to save.
- **token** – The API-token obtained via the login() method.
- **editmode** – The mode which shall be used to persist the changes. Currently "new" and "bypass" are supported.

Returns True if the action succeeded.

save_package(*object_id*, *token*, *editmode='bypass'*)
Saves a newly created or modified object to disk. Calling save is required for any changes to persist.

Parameters

- **object_id** – The id of the object to save.
- **token** – The API-token obtained via the login() method.
- **editmode** – The mode which shall be used to persist the changes. Currently "new" and "bypass" are supported.

Returns True if the action succeeded.

save_profile(*object_id*, *token*, *editmode='bypass'*)
Saves a newly created or modified object to disk. Calling save is required for any changes to persist.

Parameters

- **object_id** – The id of the object to save.
- **token** – The API-token obtained via the login() method.
- **editmode** – The mode which shall be used to persist the changes. Currently "new" and "bypass" are supported.

Returns True if the action succeeded.

save_repo(*object_id*, *token*, *editmode='bypass'*)
Saves a newly created or modified object to disk. Calling save is required for any changes to persist.

Parameters

- **object_id** – The id of the object to save.
- **token** – The API-token obtained via the login() method.
- **editmode** – The mode which shall be used to persist the changes. Currently "new" and "bypass" are supported.

Returns True if the action succeeded.

save_system(*object_id*, *token*, *editmode='bypass'*)
Saves a newly created or modified object to disk. Calling save is required for any changes to persist.

Parameters

- **object_id** – The id of the object to save.
- **token** – The API-token obtained via the login() method.
- **editmode** – The mode which shall be used to persist the changes. Currently "new" and "bypass" are supported.

Returns True if the action succeeded.

sync(*token*)
Run sync code, which should complete before XMLRPC timeout. We can't do reposync this way. Would be nice to send output over AJAX/other later.

Parameters **token** – The API-token obtained via the login() method. Cobbler token, obtained form login()

Returns bool if operation was successful

sync_dhcp(*token*)
Run sync code, which should complete before XMLRPC timeout. We can't do reposync this way. Would be nice to send output over AJAX/other later.

Parameters **token** – The API-token obtained via the login() method. Cobbler token, obtained form login()

Returns bool if operation was successful

token_check(*token*)
Checks to make sure a token is valid or not.

Parameters **token** – The API-token obtained via the login() method. Cobbler token, obtained form login()

Returns if operation was successful or not

Return type bool

upload_log_data(*sys_name*, *file*, *size*, *offset*, *data*, *token=None*, ***rest*)
This is a logger function used by the "anamon" logging system to upload all sorts of misc data from Anaconda. As it's a bit of a potential log-flooder, it's off by default and needs to be enabled in /etc/cobbler/settings.

Parameters

- **sys_name** – The name of the system for which to upload log data.
- **file** – The file where the log data should be put.
- **size** – The size of the data which will be recieved.
- **offset** – The offset in the file where the data will be written to.
- **data** – The data that should be logged.
- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns True if everything succeeded.

version (*token=None*, ***rest*)
Return the Cobbler version for compatibility testing with remote applications. See api.py for documentation.

Parameters

- **token** – The API-token obtained via the login() method.
- **rest** – This is dropped in this method since it is not needed here.

Returns The short version of Cobbler.

write_autoinstall_snippet (*file_path*, *data*, *token*)
Write an automatic OS installation snippet file

Parameters

- **file_path** (*str*) – automatic OS installation snippet file path
- **data** (*str*) – new file content
- **token** – Cobbler token, obtained form login()

Returns if operation was successful

Return type bool

write_autoinstall_template (*file_path*, *data*, *token*)
Write an automatic OS installation template file

Parameters

- **file_path** (*str*) – automatic OS installation template file path
- **data** (*str*) – new file content
- **token** – The API-token obtained via the login() method. Cobbler token, obtained form login()

Returns bool if operation was successful

xapi_object_edit (*object_type*, *object_name*, *edit_type*, *attributes*, *token*)
Extended API: New style object manipulations, 2.0 and later.

Extended API: New style object manipulations, 2.0 and later preferred over using `new_*`, `modify_*`, `save_*` directly. Though we must preserve the old ways for backwards compatibility these cause much less XMLRPC traffic.

Ex: xapi_object_edit("distro","el5","add",{"kernel":"/tmp/foo","initrd":"/tmp/foo"},token)

Parameters

- **object_type** – The object type which corresponds to the collection type the object is in.
- **object_name** – The name of the object under question.
- **edit_type** – One of 'add', 'rename', 'copy', 'remove'
- **attributes** – The attributes which shall be edited. This should be JSON-style string.
- **token** – The API-token obtained via the login() method.

Returns True if the action succeeded.

xmlrpc_hacks (*data*)
Convert None in XMLRPC to just '~' to make extra sure a client that can't allow_none can deal with this.

ALSO: a weird hack ensuring that when dicts with integer keys (or other types) are transmitted with string keys.

Parameters **data** – The data to prepare for the XMLRPC response.

Returns The converted data.

class cobbler.remote.**CobblerXMLRPCServer**(*args*)
Bases: socketserver.ThreadingMixIn, xmlrpc.server.SimpleXMLRPCServer

This is the class for the main Cobbler XMLRPC Server. This class does not directly contain all XMLRPC methods. It just starts the server.

class cobbler.remote.**ProxiedXMLRPCInterface**(*api*, *proxy_class*)
Bases: object

8.16 cobbler.resource module

An Resource is a serializable thing that can appear in a Collection

Copyright 2006-2009, Red Hat, Inc and Others Kelsey Hightower <khightower@gmail.com>

This software may be freely redistributed under the terms of the GNU general public license.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

class cobbler.resource.**Resource**(*collection_mgr*, *is_subobject=False*)
Bases: *cobbler.items.item.Item*

Base Class for management resources.

set_action(*action*)
All management resources have an action. Action determine weather a most resources should be created or removed, and if packages should be installed or uninstalled.

Parameters **action** (*str*) – The action which should be executed for the management resource. Must be on of "create" or "remove". Parameter is case-insensitive.

set_group(*group*)
Unix group ownership of a file or directory.

Parameters **group** – The group which the resource will belong to.

set_mode(*mode*)
Unix file permission mode ie: '0644' assigned to file and directory resources.

Parameters **mode** – The mode which the resource will have.

set_owner(*owner*)
Unix owner of a file or directory.

Parameters **owner** – The owner which the resource will belong to.

set_path(*path*)
File path used by file and directory resources.

Parameters **path** – Normally a absolute path of the file or directory to create or manage.

set_template(*template*)
Path to cheetah template on Cobbler's local file system. Used to generate file data shipped to koan via json. All templates have access to flatten autoinstall_meta data.

Parameters **template** – The template to use for the resource.

8.17 cobbler.serializer module

Serializer code for Cobbler Now adapted to support different storage backends

cobbler.serializer.**deserialize**(*collection*, *topological=True*)
Load a collection from disk.

Parameters

- **collection** – The Cobbler collection to know the type of the item.
- **topological** (*bool*) – Unkown parameter.

cobbler.serializer.**handler**(*num*, *frame*)

cobbler.serializer.**serialize**(*collection*)
Save a collection to disk

Parameters **collection** – The collection to serialize.

cobbler.serializer.**serialize_delete**(*collection*, *item*)
Delete a collection item from disk

Parameters

- **collection** – The Cobbler collection to know the type of the item.
- **item** – The collection item to delete.

cobbler.serializer.**serialize_item**(*collection*, *item*)
Save a collection item to disk

Parameters

- **collection** – The Cobbler collection to know the type of the item.
- **item** – The collection item to serialize.

8.18 cobbler.services module

Mod Python service functions for Cobbler's public interface (aka cool stuff that works with wget/curl)

based on code copyright 2007 Albert P. Tobey <tobert@gmail.com> additions: 2007-2009 Michael DeHaan <michael.dehaan AT gmail>

class cobbler.services.**CobblerSvc**(*server=None*, *req=None*)
Bases: object

Interesting mod python functions are all keyed off the parameter mode, which defaults to index. All options are passed as parameters into the function.

autodetect(***rest*)
This tries to autodect the system with the given information. If more than one candidate is found an error message is returned.

Parameters **rest** – The keys "REMOTE_MACS", "REMOTE_ADDR" or "interfaces".

Returns The name of the possible object or an error message.

Return type str

autoinstall(*profile=None*, *system=None*, *REMOTE_ADDR=None*, *REMOTE_MAC=None*, ***rest*)
Generate automatic installation files.

Parameters

- **profile** –
- **system** –
- **REMOTE_ADDR** –
- **REMOTE_MAC** –
- **rest** – This parameter is unused.

Returns

bootcfg(*profile=None*, *system=None*, ***rest*)
Generate a boot.cfg config file. Used primarily for VMware ESXi.

Parameters

- **profile** –
- **system** –
- **rest** – This parameter is unused.

Returns

debug(*profile=None*, ***rest*)

events(*user=''*, ***rest*)
If no user is given then all events are returned. Otherwise only event associated to a user are returned.

Parameters

- **user** – Filter the events for a given user.
- **rest** – This parameter is unused.

Returns A JSON object which contains all events.

Return type str

find_autoinstall(*system=None*, *profile=None*, ***rest*)
Find an autoinstallation for a system or a profile. If this is not known different parameters can be passed to rest to find it automatically. See "autodetect".

Parameters

- **system** – The system to find the autoinstallation for,
- **profile** – The profile to find the autoinstallation for.
- **rest** – The metadata to find the autoinstallation automatically.

Returns The autoinstall script or error message.

findks (*system=None*, *profile=None*, ***rest*)
This is a legacy function which enabled Cobbler partly to be backward compatible to 2.6.6 releases.

It should be only be used if you must. Please use find_autoinstall if possible! :param system: If you wish to find a system please set this parameter to not null. Hand over the name of it. :param profile: If you wish to find a system please set this parameter to not null. Hand over the name of it. :param rest: If you wish you can try to let Cobbler autodetect the system with the MAC address. :return: Returns the autoinstall/kickstart profile.

gpxe (*profile=None*, *system=None*, *mac=None*, ***rest*)
Generate a gPXE config

Parameters

- **profile** –
- **system** –
- **mac** –
- **rest** – This parameter is unused.

Returns

index (***args*)
Just a placeholder method as an entry point.

Parameters **args** – This parameter is unused.

Returns "no mode specified"

Return type str

ks (*profile=None*, *system=None*, *REMOTE_ADDR=None*, *REMOTE_MAC=None*, ***rest*)
Generate automatic installation files. This is a legacy function for part backward compability to 2.6.6 releases.

Parameters

- **profile** –
- **system** –
- **REMOTE_ADDR** –
- **REMOTE_MAC** –
- **rest** – This parameter is unused.

Returns

list (*what='systems'*, ***rest*)
Return a list of objects of a desired category. Defaults to "systems".

Parameters

- **what** – May be "systems", "profiles", "distros", "images", "repos", "mgmtclasses", "packages" or "files"
- **rest** – This parameter is unused.

Returns The list of object names.

Return type str

look (***rest*)
Debug only: Show the handed dict via repr to the requester. :param rest: The dict to represent. :return: The dict reformated with repr() :rtype; str

nopxe (*system=None*, ***rest*)
Disables the network boot for the given system.

Parameters

- **system** – The system to disable netboot for.
- **rest** – This parameter is unused.

Returns A boolean status if the action succeed or not.

Return type str

puppet(*hostname=None*, ***rest*)
Dump the puppet data which is available for Cobbler.

Parameters

- **hostname** – The hostname for the system which should the puppet data be dumped for.
- **rest** – This parameter is unused.

Returns The yaml for the host.

Return type str

script(*profile=None*, *system=None*, ***rest*)
Generate a script based on snippets. Useful for post or late-action scripts where it's difficult to embed the script in the response file.

Parameters

- **profile** – The profile to generate the script for.
- **system** – The system to generate the script for.
- **rest** – This may contain a parameter with the key "query_string" which has a key "script" which may be an array. The element from position zero is taken.

Returns The generated script.

Return type str

template(*profile=None*, *system=None*, *path=None*, ***rest*)
Generate a templated file for the system. Either specify a profile OR a system.

Parameters

- **profile** – The profile to provide for the generation of the template.
- **system** – The system to provide for the generation of the template.
- **path** – The path to the template.
- **rest** – This parameter is unused.

Returns The rendered template.

Return type str

trig(*mode='?'*, *profile=None*, *system=None*, *REMOTE_ADDR=None*, ***rest*)
Hook to call install triggers. Only valid for a profile OR a system.

Parameters

- **mode** (*str*) – Can be "pre", "post" or "firstboot". Everything else is invalid.
- **profile** – The profile object to run triggers for.
- **system** – The system object to run triggers for.
- **REMOTE_ADDR** – The ip if the remote system/profile.
- **rest** – This parameter is unused.

Returns The return code of the action.

Return type str

yum(*profile=None*, *system=None*, ***rest*)
Generate a repo config. Either specify a profile OR a system.

Parameters

- **profile** – The profile to provide for the generation of the template.
- **system** – The system to provide for the generation of the template.
- **rest** – This parameter is unused.

Returns The generated repository config.

Return type str

8.19 cobbler.settings module

Cobbler app-wide settings

Copyright 2006-2008, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.settings.**Settings**
Bases: object

static collection_type()
This is a hardcoded string which represents the collection type.

Returns "setting"

Return type str

static collection_types() → str
return the collection plural name

from_dict(*_dict*)
Modify this object to load values in dictionary.

WARNING: If the dict from the args has not all settings included Cobbler may behave unexpectedly.

Parameters **_dict** – The dictionary with settings to replace.

Returns Returns the settings instance this method was called from.

set(*name*, *value*)
Alias for setting an option "name" to the new value "value". (See __settattr__)

Parameters

- **name** – The name of the setting to set.
- **value** – The value of the setting to set.

Returns 0 if the action was completed successfully. No return if there is an error.

to_dict()
Return an easily serializable representation of the config.

Returns The dict with all user settings combined with settings which are left to the default.

Return type dict

to_string()
Returns the kernel options as a string.

Returns The multiline string with the kernel options.

Return type str

8.20 cobbler.templar module

Cobbler uses Cheetah templates for lots of stuff, but there's some additional magic around that to deal with snippets/etc. (And it's not spelled wrong!)

Copyright 2008-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.templar.**Templar**(*collection_mgr*, *logger=None*)
Bases: object

check_for_invalid_imports(*data*)
Ensure that Cheetah code is not importing Python modules that may allow for advanced privileges by ensuring we whitelist the imports that we allow.

Parameters **data** – The Cheetah code to check.

render(*data_input*, *search_table*, *out_path*, *subject=None*, *template_type=None*)
Render data_input back into a file.

Parameters

- **data_input** – is either a string or a filename
- **search_table** – is a dict of metadata keys and values out_path if not-none writes the results to a file (though results are always returned)
- **out_path** – Optional parameter which (if present), represents the target path to write the result into.
- **subject** – is a profile or system object, if available (for snippet eval)
- **template_type** (*str*) – May currently be "cheetah" or "jinja2".

Returns The rendered template.

Return type str

render_cheetah(*raw_data*, *search_table*, *subject=None*)
Render data_input back into a file.

Parameters

- **raw_data** – Is the template code which is not rendered into the result.
- **search_table** – is a dict of metadata keys and values (though results are always returned)
- **subject** – is a profile or system object, if available (for snippet eval)

Returns The rendered Cheetah Template.

render_jinja2(*raw_data*, *search_table*, *subject=None*)
Render data_input back into a file.

Parameters

- **raw_data** – Is the template code which is not rendered into the result.
- **search_table** – is a dict of metadata keys and values
- **subject** – is a profile or system object, if available (for snippet eval)

Returns The rendered Jinja2 Template.

cobbler.templar.**jinja2_available = True**
FIXME: log a message here

8.21 cobbler.template_api module

Cobbler provides builtin methods for use in Cheetah templates. $SNIPPET is one such function and is now used to implement Cobbler's SNIPPET:: syntax.

Written by Daniel Guernsey <danpg102@gmail.com> Contributions by Michael DeHaan <michael.dehaan AT gmail> US Government work; No explicit copyright attached to this file.

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.template_api.**Template**(***kwargs*)
Bases: Cheetah.Template.Template

This class will allow us to include any pure python builtin functions. It derives from the cheetah-compiled class above. This way, we can include both types (cheetah and pure python) of builtins in the same base template. We don't need to override __init__

SNIPPET(*file*)
Include the contents of the named snippet here. This is equivalent to the #include directive in Cheetah, except that it searches for system and profile specific snippets, and it includes the snippet's namespace.

This may be a little frobby, but it's really cool. This is a pure python portion of SNIPPET that appends the snippet's searchList to the caller's searchList. This makes any #defs within a given snippet available to the template that included the snippet.

Parameters **file** – The snippet file to read and include in the template.

Returns The updated template.

classmethod compile(**args*, ***kwargs*)
Compile a cheetah template with Cobbler modifications. Modifications include SNIPPET:: syntax replacement and inclusion of Cobbler builtin methods.

Parameters

- **args** – These just get passed right to Cheetah.
- **kwargs** – We just execute our own preprocessors and remove them and let afterwards handle Cheetah the rest.

Returns The compiled template.

Return type bytes

read_snippet(*file*)
Locate the appropriate snippet for the current system and profile and read it's contents.

This file could be located in a remote location.

This will first check for a per-system snippet, a per-profile snippet, a distro snippet, and a general snippet. If no snippet is located, it returns None.

Parameters **file** – The file to read-

Returns None (if the snippet file was not found) or the string with the read snippet.

Return type str

sedesc(*value*)
Escape a string for use in sed.

Parameters **value** – The phrase to escape.

Returns The escaped phrase.

8.22 cobbler.tftpgen module

Generate files provided by TFTP server based on Cobbler object tree. This is the code behind 'cobbler sync'.

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.tftpgen.**TFTPGen**(*collection_mgr*, *logger*)
Bases: object

Generate files provided by TFTP server

build_kernel_options(*system*, *profile*, *distro*, *image*, *arch*, *autoinstall_path*)
Builds the full kernel options line.

Parameters

- **system** – The system to generate the kernel options for.
- **profile** – Although the system contains the profile please specify it explicitly here.
- **distro** – Although the profile contains the distribution please specify it explicitly here.
- **image** – The image to generate the kernel options for.
- **arch** (*str*) – The processor architecture to generate the kernel options for.
- **autoinstall_path** – The autoinstallation path. Normally this will be a URL because you want to pass a link to an autoyast, preseed or kickstart file.

Returns The generated kernal line options.

Return type str

copy_bootloaders(*dest*)
Copy bootloaders to the configured tftpboot directory NOTE: we support different arch's if defined in /etc/cobbler/settings.

copy_images()
Like copy_distros except for images.

copy_single_distro_file(*d_file*, *distro_dir*, *symlink_ok*)
Copy a single file (kernel/initrd) to distro's images directory

Parameters

- **d_file** (*str*) – distro's kernel/initrd absolut or remote file path value
- **distro_dir** (*str*) – directory (typically in {www,tftp}/images) where to copy the file
- **symlink_ok** (*bool*) – whethere it is ok to symlink the file. Typically false in case the file is used by daemons run in chroot environments (tftpd,..)

Raises ***CX*** – Cobbler Exception is raised in case file IO errors or of the remote file could not be retrieved

Returns None

copy_single_distro_files(*d*, *dirtree*, *symlink_ok*)
Copy the files needed for a single distro.

Parameters

- **d** – The distro to copy.
- **dirtree** – This is the root where the images are located. The folder "images" gets automatically appended.
- **symlink_ok** (*bool*) – If it is okay to use a symlink to link the destination to the source.

copy_single_image_files(*img*)
Copies an image to the images directory of Cobbler.

Parameters **img** – The image to copy.

generate_bootcfg(*what*, *name*)
Generate a bootcfg for a system of profile.

Parameters

- **what** (*str*) – The type for what the bootcfg is generated for. Must be "profile" or "system".
- **name** (*str*) – The name of the item which the bootcfg should be generated for.

Returns The fully rendered bootcfg as a string.

Return type str

generate_gpxe(*what*, *name*)
Generate the gpxe files.

Parameters

- **what** (*str*) – either "profile" or "system". All other item types not valdi.
- **name** (*str*) – The name of the profile or system.

Returns The rendered template.

Return type str

generate_script(*what*, *objname*, *script_name*)
Generate a script from a autoinstall script template for a given profile or system.

Parameters

- **what** (*str*) – The type for what the bootcfg is generated for. Must be "profile" or "system".
- **objname** (*str*) – The name of the item which the bootcfg should be generated for.
- **script_name** – The name of the template which should be rendered for the system or profile.

Returns The fully rendered script as a string.

Return type str

get_menu_items(*arch=None*)

Generates menu items for pxe and grub. Grub menu items are grouped into submenus by profile.

Parameters **arch** (*str*) – The processor architecture to generate the menu items for. (Optional)

Returns A dictionary with the pxe and grub menu items. It has the keys "pxe" and "grub".

Return type dict

make_pxe_menu()

Generates both pxe and grub boot menus.

write_all_system_files(*system*, *menu_items*)

Writes all files for tftp for a given system with the menu items handed to this method. The system must have a profile attached. Otherwise this method throws an error.

Parameters

- **system** – The system to generate files for.
- **menu_items** –

write_pxe_file(*filename*, *system*, *profile*, *distro*, *arch*, *image=None*, *include_header=True*, *metadata=None*, *format='pxe'*)

Write a configuration file for the boot loader(s).

More system-specific configuration may come in later, if so that would appear inside the system object in api.py Can be used for different formats, "pxe" (default) and "grub".

Parameters

- **filename** – If present this writes the output into the giving filename. If not present this method just returns the generated configuration.
- **system** – If you supply a system there are other templates used then when using only a profile/image/ distro.
- **profile** – The profile to generate the pxe-file for.
- **distro** – If you don't ship an image, this is needed. Otherwise this just supplies information needed for the templates.
- **arch** (*str*) – The processor architecture to generate the pxefile for.
- **image** – If you want to be able to deploy an image, supply this parameter.
- **include_header** (*bool*) – Not used parameter currently.
- **metadata** – Pass additional parameters to the ones being collected during the method.
- **format** (*str*) – May be "grub" or "pxe".

Returns The generated filecontent for the required item.

Return type str

write_templates(*obj*, *write_file=False*, *path=None*)
A semi-generic function that will take an object with a template_files dict {source:destiation}, and generate a rendered file. The write_file option allows for generating of the rendered output without actually creating any files.

Parameters

- **obj** – The object to write the template files for.
- **write_file** (*bool*) – If the generated template should be written to the disk.
- **path** – TODO: A useless parameter?

Returns A dict of the destination file names (after variable substitution is done) and the data in the file.

8.23 cobbler.utils module

Misc heavy lifting functions for Cobbler

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.utils.**MntEntObj**(*input=None*)
Bases: object

mnt_dir = None

mnt_freq = 0

mnt_fsname = None

mnt_opts = None

mnt_passno = 0

mnt_type = None

cobbler.utils.**blender**(*api_handle*, *remove_dicts*, *root_obj*)
Combine all of the data in an object tree from the perspective of that point on the tree, and produce a merged dictionary containing consolidated data.

Parameters

- **api_handle** – The api to use for collecting the information to blender the item.
- **remove_dicts** – Boolean to decide whether dicts should be converted.
- **root_obj** – The object which should act as the root-node object.

Returns A dictionary with all the information from the root node downwards.

cobbler.utils.**cachefile**(*src*, *dst*, *api=None*, *logger=None*)
Copy a file into a cache and link it into place. Use this with caution, otherwise you could end up copying data twice if the cache is not on the same device as the destination.

Parameters

- **src** – The sourcefile for the copy action.

- **dst** – The destination for the copy action.
- **api** – The api to resolve basic information with.
- **logger** – The logger to audit the action with.

cobbler.utils.**cheetah_exc**(*exc*, *full=False*)
Converts an exception thrown by Cheetah3 into a custom error message.

Parameters

- **exc** – The exception to convert.
- **full** – Unused parameter. May be removed in the future.

Returns The string representation of the Cheetah3 exception.

Return type str

cobbler.utils.**clear_from_fields**(*item*, *fields*, *is_subobject=False*)
Used by various item_*.py classes for automating datastructure boilerplate.

Parameters

- **item** – The item to clear the fields of.
- **fields** – Not known what magic this actually does.
- **is_subobject** (*bool*) – If in the Cobbler inheritance tree the item is considered a subobject (True) or not (False).

cobbler.utils.**compare_versions_gt**(*ver1*, *ver2*)
Compares versions like "0.9.3" with each other and decides if ver1 is greater than ver2.

Parameters

- **ver1** – The first version.
- **ver2** – The second version.

Returns True if ver1 is greater, otherwise False.

Return type bool

cobbler.utils.**copyfile**(*src*, *dst*, *api=None*, *logger=None*)
Copy a file from source to the destination.

Parameters

- **src** – The source file. This may also be a folder.
- **dst** – The destination for the file or folder.
- **api** – This parameter is not used currently.
- **logger** – The logger to audit the action with.

cobbler.utils.**copyfile_pattern**(*pattern*, *dst*, *require_match=True*, *symlink_ok=False*, *cache=True*, *api=None*, *logger=None*)
Copy 1 or more files with a pattern into a destination.

Parameters

- **pattern** – The pattern for finding the required files.
- **dst** – The destination for the file(s) found.
- **require_match** (*bool*) – If the glob pattern does not find files should an error message be thrown or not.
- **symlink_ok** (*bool*) – If it is okay to just use a symlink to link the file to the destination.

- **cache** (*bool*) – If it is okay to use a file from the cache (which could be possibly newer) or not.
- **api** –
- **logger** – The logger to audit the action with.

cobbler.utils.**copyremotefile**(*src*, *dst1*, *api=None*, *logger=None*)
Copys a file from a remote place to the local destionation.

Parameters

- **src** – The remote file URI.
- **dst1** – The copy destination on the local filesystem.
- **api** – This parameter is not used currently.
- **logger** – The logger to audit the action with.

cobbler.utils.**dhcp_service_name**(*api*)
Determine the dhcp service which is different on various distros. This is currently a hardcoded detection.

Parameters **api** – This parameter is currently unused.

Returns This will return one of the following names: "dhcp3-server", "isc-dhcp-server", "dhcpd"

cobbler.utils.**dhcpconf_location**(*api*)
This method returns the location of the dhcpd.conf file.

Parameters **api** – This parameter is currently unused.

Returns The path possibly used for the dhcpd.conf file.

cobbler.utils.**dict_removals**(*results*, *subkey*)
Remove entrys from a dictionary starting with a "!".

Parameters

- **results** – The dictionary to search in
- **subkey** – The subkey to search through.

cobbler.utils.**dict_to_string**(*_dict*)
Convert a dictionary to a printable string. Used primarily in the kernel options string and for some legacy stuff where koan expects strings (though this last part should be changed to dictionaries)

A KV-Pair is joined with a "=". Values are enclosed in single quotes.

Parameters **_dict** – The dictionary to convert to a string.

Returns The string which was previously a dictionary.

Return type str

cobbler.utils.**die**(*logger*, *msg*)
This method let's Cobbler crash with an exception. Log the exception once in the per-task log or the main log if this is not a background op.

Parameters

- **logger** –
- **msg** –

cobbler.utils.**file_is_remote**(*file_location*)
Returns true if the file is remote and referenced via a protocol we support.

Parameters **file_location** – The URI to check.

Returns True if the URI is http, https or ftp. Otherwise false.

Return type bool

cobbler.utils.**find_distro_path**(*settings*, *distro*)
This returns the absolute path to the distro under the `distro_mirror` directory. If that directory doesn't contain the kernel, the directory of the kernel in the distro is returned.

Parameters

- **settings** – The settings to resolve user configurable actions with.
- **distro** – The distribution to find the path of.

Returns The path to the distribution files.

cobbler.utils.**find_highest_files**(*directory*, *unversioned*, *regex*)
Find the highest numbered file (kernel or initrd numbering scheme) in a given directory that matches a given pattern. Used for auto-booting the latest kernel in a directory.

Parameters

- **directory** – The directory to search in.
- **unversioned** – The base filename which also acts as a last resort if no numbered files are found.
- **regex** – The regex to search for.

Returns None or the file with the highest number.

cobbler.utils.**find_initrd**(*path*)
Given a directory or a filename, see if the path can be made to resolve into an intird, return that full path if possible.

Parameters **path** – The path to check for initrd files.

Returns None or the path to the found initrd.

cobbler.utils.**find_kernel**(*path*)
Given a directory or a filename, find if the path can be made to resolve into a kernel, and return that full path if possible.

Parameters **path** – The path to check for a kernel.

Returns None or the path with the kernel.

cobbler.utils.**find_matching_files**(*directory*, *regex*)
Find all files in a given directory that match a given regex. Can't use glob directly as glob doesn't take regexen. The search does not include subdirectories.

Parameters

- **directory** – The directory to search in.
- **regex** – The regex to apply to the found files.

Returns An array of files which apply to the regex.

Return type list

cobbler.utils.**flatten**(*data*)
Convert certain nested dicts to strings. This is only really done for the ones koan needs as strings this should not be done for everything

Parameters **data** – The dictionary in which various keys should be converted into a string.

Return type dict

Returns None (if data is None) or the flattened string.

Return type None or dict

cobbler.utils.**from_dict_from_fields**(*item*, *item_dict*, *fields*)
Not known what this method does exactly.

Parameters

- **item** – Not known what this is needed for exactly.
- **item_dict** – Not known what this is needed for exactly.
- **fields** – Not known what this is needed for exactly.

cobbler.utils.**get_exc**(*exc*, *full=True*)
This tries to analyze if an exception comes from Cobbler and potentially enriches or shortens the exception.

Parameters

- **exc** – The exception which should be analyzed.
- **full** – If the full exception should be returned or only the most important information.

Returns The exception which has been converted into a string which then can be logged easily.

cobbler.utils.**get_family**()
Get family of running operating system.

Family is the base Linux distribution of a Linux distribution, with a set of common parents.

Returns May be "redhat", "debian" or "suse" currently. If none of these are detected then just the distro name is returned.

Return type str

cobbler.utils.**get_file_device_path**(*fname*)

What this function attempts to do is take a file and return:

- the device the file is on
- the path of the file relative to the device.

For example: /boot/vmlinuz -> (/dev/sda3, /vmlinuz) /boot/efi/efi/redhat/elilo.conf -> (/dev/cciss0, /elilo.conf) /etc/fstab -> (/dev/sda4, /etc/fstab)

Parameters **fname** – The filename to split up.

Returns A tuple containing the device and relative filename.

cobbler.utils.**get_host_ip**(*ip*, *shorten=True*)
Return the IP encoding needed for the TFTP boot tree.

Parameters

- **ip** – The IP address to pretty print.
- **shorten** – Whether the IP-Address should be shortened or not.

Return type str

cobbler.utils.**get_mtab**(*mtab='/etc/mtab'*, *vfstype=None*)
Get the list of mtab entries. If a custum mtab should be read then the location can be overriden via a parameter.

Parameters

- **mtab** – The location of the mtab. Argument can be ommited if the mtab is at its default location.
- **vfstype** (*bool*) – If this is True, then all filesystems which are nfs are returned. Otherwise this returns all mtab entries.

Returns The list of requested mtab entries.

Return type list

cobbler.utils.**get_random_mac**(*api_handle*, *virt_type='xenpv'*)
Generate a random MAC address.

The code of this method was taken from xend/server/netif.py

Parameters

- **api_handle** – The main Cobbler api instance.
- **virt_type** – The virtualization provider. Currently possible is 'vmware', 'xen', 'qemu', 'kvm'.

Returns MAC address string

Return type str

cobbler.utils.**get_setter_methods_from_fields**(*item*, *fields*)
Return the name of set functions for all fields, keyed by the field name.

Parameters

- **item** – The item to search for setters.
- **fields** – The fields to search for setters.

Returns The dictionary with the setter methods.

cobbler.utils.**get_shared_secret**()
The 'web.ss' file is regenerated each time cobblerd restarts and is used to agree on shared secret interchange between the web server and cobblerd, and also the CLI and cobblerd, when username/password access is not required. For the CLI, this enables root users to avoid entering username/pass if on the Cobbler server.

Returns The Cobbler secret which enables full access to Cobbler.

Return type str

cobbler.utils.**get_supported_distro_boot_loaders**(*distro*, *api_handle=None*)
This is trying to return you the list of known bootloaders if all resorts fail. Otherwise this returns a list which contains only the subset of bootloaders which are available by the distro in the argument.

Parameters

- **distro** – The distro to check for.
- **api_handle** – The api instance to resolve metadata and settings from.

Returns The list of bootloaders or a dict of well known bootloaders.

cobbler.utils.**get_supported_system_boot_loaders**()
Return the list of currently supported bootloaders.

Returns The list of currently supported bootloaders.

Return type list

cobbler.utils.**get_valid_archs**()
Return a list of valid architectures found in the import signatures

Returns All architectures which are known to Cobbler according to the signature cache.

cobbler.utils.**get_valid_breeds**()
Return a list of valid breeds found in the import signatures

Return type list

cobbler.utils.**get_valid_os_versions**()
Return a list of valid os-versions found in the import signatures

Returns All operating system versions which are known to Cobbler according to the signature cache.

Return type list

cobbler.utils.**get_valid_os_versions_for_breed**(*breed*)
Return a list of valid os-versions for the given breed

Parameters **breed** – The operating system breed to check for.

Returns All operating system version which are known to Cobbler according to the signature cache filtered by a os-breed.

Return type list

cobbler.utils.**grab_tree**(*api_handle*, *item*)
Climb the tree and get every node.

Parameters

- **api_handle** – The api to use for checking the tree.
- **item** – The item to check for parents

Returns The list of items with all parents from that object upwards the tree. Contains at least the item itself.

Return type list

cobbler.utils.**hashfile**(*fn*, *lcache=None*, *logger=None*)
Returns the sha1sum of the file

Parameters

- **fn** – The file to get the sha1sum of.
- **lcache** – Not known what this is exactly for.
- **logger** – The logger to audit the action with.

Returns The sha1 sum or None if the file doesn't exist.

cobbler.utils.**input_boolean**(*value*)
Convert a str to a boolean. If this is not possible or the value is false return false.

Parameters **value** (*str*) – The value to convert to boolean.

Returns True if the value is in the following list, otherwise false: "true", "1", "on", "yes", "y" .

Return type bool

cobbler.utils.**input_string_or_dict**(*options*, *allow_multiples=True*)
Older Cobbler files stored configurations in a flat way, such that all values for strings. Newer versions of Cobbler allow dictionaries. This function is used to allow loading of older value formats so new users of Cobbler aren't broken in an upgrade.

Parameters

- **options** – The str or dict to convert.
- **allow_multiples** – True (default) to allow multiple identical keys, otherwise set this false explicitly.

Returns A tuple of True and a dict.

cobbler.utils.**input_string_or_list**(*options*)
Accepts a delimited list of stuff or a list, but always returns a list.

Parameters **options** – The object to split into a list.

Returns str when this functions get's passed <<inherit>>. if option is delete then an empty list is returned. Otherwise this function tries to return the arg option or tries to split it into a list.

Return type list or str

cobbler.utils.**is_ip**(*strdata*)
Return whether the argument is an IP address.

Parameters **strdata** (*str*) – The IP in a string format. This get's passed to the IP object of Python.

Return type bool

cobbler.utils.**is_mac**(*strdata*)
Return whether the argument is a mac address.

Return type bool

cobbler.utils.**is_remote_file**(*file*)
This function is trying to detect if the file in the argument is remote or not.

Parameters **file** – The filepath to check.

Returns If remote True, otherwise False.

Return type bool

cobbler.utils.**is_safe_to_hardlink**(*src*, *dst*, *api*)
Determine if it is safe to hardlink a file to the destination path.

Parameters

- **src** – The hardlink source path.
- **dst** – The hardlink target path.
- **api** – The api-instance to resolve needed information with.

Returns True if selinux is disabled, the file is on the same device, the source in not a link, and it is not a remote path. If selinux is enabled the functions still may return true if the object is a kernel or initrd. Otherwise returns False.

Return type bool

cobbler.utils.**is_selinux_enabled**()
This check is achieved via a subprocess call to `selinuxenabled`. Default return is false.

Returns Whether selinux is enabled or not.

Return type bool

cobbler.utils.**is_systemd**()
Return whether or not this system uses systemd.

This method currently checks if the path /usr/lib/systemd/systemd exists.

Return type bool

cobbler.utils.**kopts_overwrite**(*system*, *distro*, *kopts*, *settings*)
SUSE is not using 'text'. Instead 'textmode' is used as kernel option.

Parameters

- **system** – The system to overwrite the kopts for.
- **distro** – The distro for the system to change to kopts for.
- **kopts** – The kopts of the system.
- **settings** – The settings instance of Cobbler.

cobbler.utils.**link_distro**(*settings*, *distro*)
Link a Cobbler distro from its source into the web directory to make it reachable from the outside.

Parameters

- **settings** – The settings to resolve user configurable actions with.
- **distro** – The distro to link into the Cobbler web directory.

cobbler.utils.**linkfile**(*src*, *dst*, *symlink_ok=False*, *cache=True*, *api=None*, *logger=None*)
Attempt to create a link dst that points to src. Because file systems suck we attempt several different methods or bail to just copying the file.

Parameters

- **src** – The source file.

- **dst** – The destination for the link.
- **symlink_ok** (*bool*) – If it is okay to just use a symbolic link.
- **cache** (*bool*) – If it is okay to use a cached file instead of the real one.
- **api** – This parameter is needed to check if a file can be hardlinked. This method fails if this parameter is not present.
- **logger** – If a logger instance is present, then it is used to audit what this method is doing to the filesystem.

cobbler.utils.**load_signatures**(*filename*, *cache=True*)
Loads the import signatures for distros.

Parameters

- **filename** – Loads the file with the given name.
- **cache** (*bool*) – If the cache should be set with the newly read data.

cobbler.utils.**local_get_cobbler_api_url**()
Get the URL of the Cobbler HTTP API from the Cobbler settings file.

Returns The api entry point. This does not respect modifications from Loadbalancers or API-Gateways.

Return type str

cobbler.utils.**local_get_cobbler_xmlrpc_url**()
Get the URL of the Cobbler XMLRPC API from the Cobbler settings file.

Returns The api entry point.

Return type str

cobbler.utils.**lod_sort_by_key**(*_list*, *indexkey*)
Sorts a list of dictionaries by a given key in the dictionaries.

Note: This is a destructive operation and does not sort the dictionaries.

Parameters

- **_list** (*list*) – The list of dictionaries to sort.
- **indexkey** – The key to index to dicts in the list.

Returns The sorted list.

Return type list

cobbler.utils.**lod_to_dod**(*_list*, *indexkey*)
things like get_distros() returns a list of a dictionaries convert this to a dict of dicts keyed off of an arbitrary field

EX: [{ “a” : 2 }, { “a : 3 }] -> { “2” : { “a” : 2 }, “3” : { “a” : “3” }

Parameters

- **_list** (*list*) – The list of dictionaries to use for the conversion.
- **indexkey** (*int*) – The position to use as dictionary keys.

Returns The converted dictionary. It is not guaranteed that the same key is not used multiple times.

Return type dict

cobbler.utils.**log_exc**(*logger*)
Log an exception.

Parameters **logger** – The logger to audit all action.

cobbler.utils.**mkdir**(*path*, *mode=493*, *logger=None*)
Create directory with a given mode.

Parameters

- **path** – The path to create the directory at.
- **mode** – The mode to create the directory with.
- **logger** – The logger to audit the action with.

cobbler.utils.**mycmp**(*x*, *y*)
Compares if x is smaller than y.

Parameters

- **x** – The first parameter.
- **y** – The second parameter.

Returns The bool of the action.

Return type bool

cobbler.utils.**named_service_name**(*api*, *logger=None*)
Determine the named service which is normally different on various distros.

Parameters

- **api** – This parameter is currently unused.
- **logger** – The logger to audit the action with.

Returns This will return for debian/ubuntu bind9 and on other distros named-chroot or named.

Return type str

cobbler.utils.**namedconf_location**(*api*)
This returns the location of the named.conf file.

Parameters **api** – This parameter is currently unused.

Returns If the distro is Debian/Ubuntu then this returns "/etc/bind/named.conf". Otherwise "/etc/named.conf"

cobbler.utils.**os_release**()
Get the os version of the linux distro. If the get_family() method succeeds then the result is normalized.

Returns The os-name and os version.

cobbler.utils.**path_tail**(*apath*, *bpath*)
Given two paths (B is longer than A), find the part in B not in A

Parameters

- **apath** – The first path.
- **bpath** – The second path.

Returns If the paths are not starting at the same location this function returns an empty string.

Return type str

cobbler.utils.**pretty_hex**(*ip*, *length=8*)
Pads an IP object with leading zeroes so that the result is _length_ hex digits. Also do an upper().

Parameters

- **ip** – The IP address to pretty print.
- **length** – The length of the resulting hexstring. If the number is smaller than the resulting hex-string then no front-padding is done.

Return type str

cobbler.utils.**read_file_contents**(*file_location*, *logger=None*, *fetch_if_remote=False*)
Reads the contents of a file, which could be referenced locally or as a URI.

Parameters

- **file_location** – The location of the file to read.
- **logger** – The logger to autdit this action with.
- **fetch_if_remote** – If True a remote file will be tried to read, otherwise remote files are skipped and None is returned.

Returns Returns None if file is remote and templating of remote files is disabled.

Return type str or None

Raises ***FileNotFoundException*** – if the file does not exist at the specified location.

cobbler.utils.**remote_file_exists**(*file_url*)
Return True if the remote file exists.

Parameters **file_url** – The URL to check.

Returns True if Cobbler can reach the specified URL, otherwise false.

Return type bool

cobbler.utils.**remove_yum_olddata**(*path*, *logger=None*)
Delete .olddata files that might be present from a failed run of createrepo. # FIXME: verify this is still being used

Parameters

- **path** – The path to check for .olddata files.
- **logger** – The logger to audit this action with.

cobbler.utils.**revert_strip_none**(*data*)
Does the opposite to strip_none. If a value which represents None is detected, it replaces it with None.

Parameters **data** – The data to check.

Returns The data without None.

cobbler.utils.**rmfile**(*path*, *logger=None*)
Delete a single file.

Parameters

- **path** – The file to delete.
- **logger** – The logger to audit the action with.

Returns True if the action succeeded.

Return type bool

cobbler.utils.**rmtree**(*path*, *logger=None*)
Delete a complete directory or just a single file.

Parameters

- **path** – The directory or folder to delete.
- **logger** – The logger to audit the action with.

Returns May possibly return true on success or may return None on success.

cobbler.utils.**rmtree_contents**(*path*, *logger=None*)
Delete the content of a folder with a glob pattern.

Parameters

- **path** – This parameter presents the glob pattern of what should be deleted.

- **logger** – The logger to audit the action with.

cobbler.utils.**rsync_files**(*src*, *dst*, *args*, *logger=None*, *quiet=True*)
Sync files from src to dst. The extra arguments specified by args are appended to the command.

Parameters

- **src** – The source for the copy process.
- **dst** – The destination for the copy process.
- **args** – The extra arguments are appended to our standard arguements.
- **logger** – The logger to audit the action with.
- **quiet** (*bool*) – If "True" no progress is reported. If "False" then progress will be reported by rsync.

Returns `True` on success, otherwise `False`.

cobbler.utils.**run_this**(*cmd*, *args*, *logger*)
A simple wrapper around subprocess calls.

Parameters

- **cmd** – The command to run in a shell process.
- **args** – The arguments to attach to the command.
- **logger** – The logger to audit the shell call with.

cobbler.utils.**run_triggers**(*api*, *ref*, *globber*, *additional=[]*, *logger=None*)
Runs all the trigger scripts in a given directory. Example: `/var/lib/cobbler/triggers/blah/*`

As of Cobbler 1.5.X, this also runs Cobbler modules that match the globbing paths.

Python triggers are always run before shell triggers.

Parameters

- **api** – The api object to use for resolving the actions.
- **ref** – Can be a Cobbler object, if not None, the name will be passed to the script. If ref is None, the script will be called with no argumenets.
- **globber** – is a wildcard expression indicating which triggers to run.
- **additional** (*list*) – Additional arguments to run the triggers with.
- **logger** – The logger to audit the action with.

cobbler.utils.**safe_filter**(*var*)
Not know what this function exactly does.

Parameters **var** – This parameter shall not be None or have ".."/";" at the end.

cobbler.utils.**set_arch**(*self*, *arch*, *repo=False*)
This is a setter for system architectures. If the arch is not valid then an exception is raised.

Parameters

- **self** – The object where the arch will be set.
- **arch** – The desired architecture to set for the object.
- **repo** (*bool*) – If the object where the arch will be set is a repo or not.

cobbler.utils.**set_breed**(*self*, *breed*)
This is a setter for the operating system breed.

Parameters

- **self** – The object to set the os-breed for.
- **breed** – The os-breed which shall be set.

cobbler.utils.**set_os_version**(*self*, *os_version*)
This is a setter for the operating system version of an object.

Parameters

- **self** – The object to set the os-version for.
- **os_version** – The version which shall be set.

cobbler.utils.**set_repo_breed**(*self*, *breed*)
This is a setter for the repository breed.

Parameters

- **self** – The object to set the breed of.
- **breed** – The new value for breed.

cobbler.utils.**set_repo_os_version**(*self*, *os_version*)
This is a setter for the os-version of a repository.

Parameters

- **self** – The repo to set the os-version for.
- **os_version** – The os-version which should be set.

cobbler.utils.**set_repos**(*self*, *repos*, *bypass_check=False*)
This is a setter for the repository.

Parameters

- **self** – The object to set the repositories of.
- **repos** – The repositories to set for the object.
- **bypass_check** (*bool*) – If the newly set repos should be checked for existence.

cobbler.utils.**set_serial_baud_rate**(*self*, *baud_rate*)
The baud rate is very import that the communication between the two devices can be established correctly. This is the setter for this parameter. This effectively is the speed of the connection.

Parameters

- **self** – The object to set the serial baud rate for.
- **baud_rate** (*int*) – The baud rate to set.

Returns True if the action succeeded.

Return type bool

cobbler.utils.**set_serial_device**(*self*, *device_number*)
Set the serial device for an object.

Parameters

- **self** – The object to set the device number for.
- **device_number** (*int*) – The number of the serial device.

Returns True if the action succeeded.

Return type bool

cobbler.utils.**set_virt_auto_boot**(*self*, *num*)
For Virt only. Specifies whether the VM should automatically boot upon host reboot 0 tells Koan not to auto_boot virtuals.

Parameters

- **self** – The object where the virt auto boot should be set for.
- **num** (*int*) – May be “0” (disabled) or “1” (enabled)

cobbler.utils.**set_virt_bridge**(*self*, *vbridge*)
The default bridge for all virtual interfaces under this profile.

Parameters

- **self** – The object to adjust the virtual interfaces of.
- **vbridge** – The bridgename to set for the object.

cobbler.utils.**set_virt_cpus**(*self*, *num*)
For Virt only. Set the number of virtual CPUs to give to the virtual machine. This is fed to virtinst RAW, so Cobbler will not yelp if you try to feed it 9999 CPUs. No formatting like 9,999 please :)

Parameters

- **self** – The object to adjust the virtual cpu cores.
- **num** – The number of cpu cores.

cobbler.utils.**set_virt_disk_driver**(*self*, *driver*)
For Virt only. Specifies the on-disk format for the virtualized disk

Parameters

- **self** – The object where the virt disk driver should be set for.
- **driver** – The virt driver to set.

cobbler.utils.**set_virt_file_size**(*self*, *num*)
For Virt only: Specifies the size of the virt image in gigabytes. Older versions of koan (x<0.6.3) interpret 0 as "don't care". Newer versions (x>=0.6.4) interpret 0 as "no disks"

Parameters

- **self** – The object where the virt file size should be set for.
- **num** – is a non-negative integer (0 means default). Can also be a comma seperated list – for usage with multiple disks

cobbler.utils.**set_virt_path**(*self*, *path*, *for_system=False*)
Virtual storage location suggestion, can be overriden by koan.

Parameters

- **self** – The object to adjust the virtual storage location.
- **path** – The path to the storage.
- **for_system** (*bool*) – If this is set to True then the value is inherited from a profile.

cobbler.utils.**set_virt_pxe_boot**(*self*, *num*)
For Virt only. Specifies whether the VM should use PXE for booting 0 tells Koan not to PXE boot virtuals

Parameters

- **self** – The object where the virt pxe boot should be set for.
- **num** (*int*) – May be "0" (disabled) or "1" (enabled)

cobbler.utils.**set_virt_ram**(*self*, *num*)
For Virt only. Specifies the size of the Virt RAM in MB.

Parameters

- **self** – The object where the virtual RAM should be set for.
- **num** (*int*) – 0 tells Koan to just choose a reasonable default.

cobbler.utils.**set_virt_type**(*self*, *vtype*)
Virtualization preference, can be overridden by koan.

Parameters

- **self** – The object where the virtual machine type should be set for.

- **vtype** – May be one of “qemu”, “kvm”, “xenpv”, “xenfv”, “vmware”, “vmwarew”, “openvz” or “auto”

cobbler.utils.**strip_none**(*data*, *omit_none=False*)
Remove “None” entries from datastructures. Used prior to communicating with XMLRPC.

Parameters

- **data** – The data to strip None away.
- **omit_none** (*bool*) – If the datastructure is not a single item then None items will be skipped instead of replaced if set to “True”.

Returns The modified data structure without any occurrence of None.

cobbler.utils.**subprocess_call**(*logger*, *cmd*, *shell=True*, *input=None*)
A simple subprocess call with no output capturing.

Parameters

- **logger** – The logger to audit the action with.
- **cmd** – The command to execute.
- **shell** (*bool*) – Whether to use a shell or not for the execution of the commmand.
- **input** – If there is any input needed for that command to stdin.

Returns The return code of the process

cobbler.utils.**subprocess_get**(*logger*, *cmd*, *shell=True*, *input=None*)
A simple subprocess call with no return code capturing.

Parameters

- **logger** – The logger to audit the action with.
- **cmd** – The command to execute.
- **shell** (*bool*) – Whether to use a shell or not for the execution of the commmand.
- **input** – If there is any input needed for that command to stdin.

Returns The data which the subprocess returns.

cobbler.utils.**subprocess_sp**(*logger*, *cmd*, *shell=True*, *input=None*)
Call a shell process and redirect the output for internal usage.

Parameters

- **logger** – The logger to audit the action with.
- **cmd** – The command to execute in a subprocess call.
- **shell** (*bool*) – Whether to use a shell or not for the execution of the commmand.
- **input** – If there is any input needed for that command to stdin.

Returns A tuple of the output and the return code.

cobbler.utils.**to_dict_from_fields**(*item*, *fields*)
Not known what this method does exactly.

Parameters

- **item** – Not known what this is needed for exactly.
- **fields** – Not known what this is needed for exactly.

Returns Returns a dictionary of the fields of an item (distro, profile,..).

Return type dict

cobbler.utils.**to_string_from_fields**(*item_dict*, *fields*, *interface_fields=None*)
item_dict is a dictionary, fields is something like item_distro.FIELDS

Parameters

- **item_dict** – Not known what this is needed for exactly.
- **fields** – Not known what this is needed for exactly.
- **interface_fields** – Not known what this is needed for exactly.

Returns Not known what this is returning exactly.

Return type str

cobbler.utils.**uniquify**(*seq*)
Remove duplicates from the sequence handed over in the args.

Parameters **seq** – The sequence to check for duplicates.

Returns The list without duplicates.

Return type list

cobbler.utils.**update_settings_file**(*data*)
Write data handed to this function into the settings file of Cobbler. This function overwrites the existing content.

Parameters **data** – The data to put into the settings file.

Returns True if the action succeeded. Otherwise return nothing.

Return type bool

cobbler.utils.**zonefile_base**(*api*)
This determines the base directory for the zone files which are important for the named service which Cobbler tries to configure.

Parameters **api** – This parameter is currently unused.

Returns One of "/etc/bind/db.", "/var/lib/named/", "/var/named/". The result depends on the distro used.

8.24 cobbler.validate module

cobbler.validate.**hostname**(*dnsname*)
Validate the dns name.

Parameters **dnsname** (*str*) – Hostname or FQDN

Returns dnsname

Raises **CX** – If the Hostname/FQDN is not a string or in an invalid format.

Return type str

cobbler.validate.**ipv4_address**(*addr*)
Validate an IPv4 address.

Parameters **addr** (*str*) – (ipv4 address)

Returns str addr or CX

cobbler.validate.**ipv4_netmask**(*addr*)
Validate an IPv4 netmask.

Parameters **addr** (*str*) – ipv4 netmask

Returns str addr or CX

cobbler.validate.**ipv6_address**(*addr*)
Validate an IPv6 address.

Parameters **addr** (*str*) – ipv6 address

Returns The ipv6 address.

Return type str

cobbler.validate.**mac_address**(*mac*, *for_item=True*)
Validate as an Eternet mac address.

Parameters **mac** (*str*) – mac address

Returns str mac or CX

cobbler.validate.**name_servers**(*nameservers*, *for_item=True*)
Validate nameservers IP addresses, works for IPv4 and IPv6

Parameters

- **nameservers** (*str or list*) – (string or list of nameserver addresses)
- **for_item** (*bool*) – (enable/disable special handling for Item objects)

Returns The list of valid nameservers.

cobbler.validate.**name_servers_search**(*search*, *for_item=True*)
Validate nameservers search domains.

Parameters

- **search** (*str or list*) – One or more search domains to validate.
- **for_item** (*bool*) – (enable/disable special handling for Item objects)

Returns The list of valid nameservers.

cobbler.validate.**object_name**(*name*, *parent*)
Validate the object name.

Parameters

- **name** (*str*) – object name
- **parent** (*str*) – Parent object name

Returns name or CX

Return type str

8.25 cobbler.yumgen module

Builds out filesystem trees/data based on the object tree. This is the code behind 'cobbler sync'.

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program 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 2 of the License, or (at your option) any later version.

This program 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 this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.yumgen.**YumGen**(*collection_mgr*)

Bases: object

get_yum_config(*obj*, *is_profile*)

Return one large yum repo config blob suitable for use by any target system that requests it.

Parameters

- **obj** – The object to generate the yumconfig for.
- **is_profile** (*bool*) – If the requested object is a profile. (Parameter not used currently)

Returns The generated yumconfig or the errors.

Return type str

8.26 Module contents

CHAPTER 9

Release Notes for Cobbler 3.0.0

9.1 Enhancements

- Use new dracut ip option for configuring static interfaces (koan).
- Add a whitelist of directories in order to persist a `cobbler sync`.
- Add proxy support for get-loaders, signature update and reposync.
- Add initial support for DJBDNS.
- Enable external YUM repo mirroring through a proxy server.
- DHCP configuration now also supports the per interface gateway setting.
- A new interface_type `BMC` was added which also can be managed with DHCP.
- Yaboot was updated to 1.3.17.
- Add ability to have per-profile/per-system `next_server` values (#1196).
- Add `--graphics` option to Koan.
- Improved input validation and error handling.
- Support `virtio26` for generic QEMU fallback in Koan.
- Debian network config: add support for tagged vlan only bonding interfaces.
- Documentation has been converted into rST and is now included with the source tree.
- Integrated pyflakes into the build system and resolved hundreds of issues.
- Integrated pep8 (coding style) into the build system and resolved thousands of issues.
- Add a new field to the system type `ipv6_prefix` (#203).
- Minor update to CSS; make better use of screen (tables) (cobbler-web).
- Add support for an empty system status.
- If `dns-name` is specified, set it as DHCP hostname in preference to the `hostname` field.
- Allow user to choose whether or not to delete item(s) recursively (cobbler-web).
- Set ksdevice kernel option to MAC address for ppc systems as bootif is not used by yaboot.

- Return to list of snippets/kickstarts when snippet/kickstart is saved (cobbler-web).
- Layout in snippet/kickstart edit form has been improved (cobbler-web).
- Better handling of copy/remove actions for subprofiles (API and cobbler-web).
- Make kickstart selectable from a pulldown list in cobbler-web (#991).

9.2 Bugfixes

- Changed Apache configuration directory in Ubuntu 14.04 (#1208).
- build_reporting no longer fails with an empty string in ignorelist (#1248).
- Kickstart repo statement, filter invalid values: `gpgcheck`, `gpgkey` and `enabled` (#323).
- Several improvements to Debian/Ubuntu packaging.
- Some class/method names have been changed to make the code more intuitive for developers.
- Remove `root=` argument in Koan when using grubby and replace-self to avoid booting the current OS.
- Exit with an error if the cobblerd executable can't be found (#1108, #1135).
- Fix cobbler sync bug by xmlrpclib returning NoneType object.
- Dont send the Puppet environment when system status is empty (#560).
- Cobbler-web kept only the most recent interface change (#687).
- Fix broken gitdate, gitstamp values in `/etc/cobbler/version`.
- Prevent disappearing profiles after cobblerd restart (#1030).
- Add missing icons to cobbler_web/content (#679).
- cobbler-ext-nodes was broken with `mgmt_classes` defined at the profile level (#790).
- Properly name the VLAN interface in the manual page.
- Fix wrong address of the Free Software Foundation.
- Remove legacy (EL5/6) cruft from the RPM specfile.
- Koan: use the print function instead of the print statement.
- Minor improvement to LDAP configuration (#217).
- Improvements to the unittest framework.
- Removed several unused functions from utils.
- List of authors is now automagically generated.

9.3 Upgrade notes

- Support for LDAP configuration through Koan has been removed.
- Support for redhat_management (Spacewalk/Satelite) has been moved to contrib. Users of this functionality should checkout contrib/redhat-management/README.
- Monit support has been removed; you really need to use a CMS to manage your services.
- Support for remote kickstart templates and files been removed (eg. kickstart=http://).
- All object names are now validated like that of the system object.
- The use of `parent` and `distro` on subprofiles are now mutually exclusive.
- Support for s390/s390x has been removed.

- Support for ia64 (Itanium) has been removed.
- Support for the MySQL backend has been removed.
- Support for deprecated fieldnames (`subnet`, `bonding_master`, `bonding`) has been removed.
- Cobbler now requires python 2.7 and Koan now requires python 2.6.
- Red Hat specific default kernel options have been removed from the settings file.
- Support for Func integration has been moved to contrib. Users of this functionality should checkout contrib/func/README.
- Deprecated Koan LiveCD: moved to contrib.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z