Contents

1 Introduction 1
   1.1 Availability ......................................................... 1
   1.2 Motivation ......................................................... 1
   1.3 How does FAI work ................................................. 2
   1.4 Features .......................................................... 2
   1.5 Installation times .................................................. 3

2 Quickstart - For the impatient user 3
   2.1 My first installation ................................................. 3
   2.2 My first server installation ....................................... 3

3 Overview and Concepts 4
   3.1 Important Terms .................................................... 4
   3.2 The class concept .................................................... 6

4 Setup your faiserver 7
   4.1 Install the FAI packages ......................................... 7
   4.2 Create the nfsroot .................................................. 7
   4.3 Creating the configuration space ................................... 8
   4.4 Configure the network daemons .................................... 9
      4.4.1 Configuration of the DHCP daemon ......................... 9
      4.4.2 Adding a host entry to DHCP ................................ 9
      4.4.3 TFTP ............................................................ 10
      4.4.4 NFS ............................................................ 10
   4.5 Creating the PXELINUX configuration ............................. 10
   4.6 Custom server ...................................................... 10

5 Plan your installation 11
   5.1 The configuration space and its subdirectories ................. 11
   5.2 Defining classes .................................................... 12
   5.3 Defining variables .................................................. 13
   5.4 Hard disk configuration ............................................. 13
   5.5 Extract base file .................................................... 14
   5.6 Debconf preseeding ................................................ 14
   5.7 Access to the package repository ................................ 14
   5.8 Software package configuration .................................. 14
   5.9 Customization scripts ............................................. 16
      5.9.1 Shell scripts .................................................. 16
      5.9.2 Cfengine scripts .............................................. 16
   5.10 Hooks ............................................................... 16
   5.11 FAI flags ............................................................ 17
6  FAI installs your plan 18
   6.1  The early part of an installation 18
   6.2  Boot messages 18
   6.3  Rebooting the computer into the new system 19
   6.4  Starting FAI (task confdir) 20
   6.5  Defining classes and variables (tasks defclass and defvar) 20
   6.6  Partitioning local disks, creating file systems (task partition) 20
   6.7  Debconf preseeding (task debconf) 20
   6.8  Installing software packages (task instsoft) 20
   6.9  Site specific customization (task configure) 20
   6.10  Saving log files (task savelog) 21
   6.11  Reboot the new installed system 21

7  Advanced FAI topics 21
   7.1  Checking parameters received from DHCP servers 21
   7.2  Monitoring multiple client installations 21
   7.3  Collecting Ethernet addresses for multiple hosts 22
       7.3.1  Debugging the network traffic 22
   7.4  Details of PXE booting 22
   7.5  Customizing your install server setup 23
   7.6  Creating a FAI CD or and USB stick 24
   7.7  Creating VM disk images using FAI 24
   7.8  Building cross-architecture disk images 24
   7.9  FAI rescue system 25
   7.10  FAI without NFS 26
   7.11  Installing other distributions using a Debian nfsroot 26
   7.12  Creating chroot and virtualization environments 26
   7.13  Using FAI for updates 27
   7.14  How to install 32-bit OS from a 64-bit OS 27
   7.15  Aborting the installation when an error occurs 28

8  Various hints and details 29
   8.1  The list of tasks 29
   8.2  Automated tests 30
   8.3  Autodiscover 30
   8.4  Changing the boot device 31
   8.5  How to use NFS v4 with the nfsroot 31
   8.6  How to create a local Debian mirror 31
   8.7  Small hints 32
   8.8  flag_reboot (FAI_FLAGS) 33
   8.9  CentOS reboot 33
   8.10  Log files 33
   8.11  How to use HTTP for PXE boot 34
9 Troubleshooting
  9.1 Boot errors
Abstract

FAI is a non-interactive system to install, customize and manage Linux systems and software configurations on computers as well as virtual machines and chroot environments, from small networks to large infrastructures and clusters.

This manual describes the Fully Automatic Installation software. This includes the installation of the packages, setting up the server, creating of the configuration and how to deal with errors.

(c) 2000-2020 Thomas Lange

Copyright This manual is free software; you may 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, or (at your option) any later version.

This 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.

A copy of the GNU General Public License is available as `/usr/share/common-licenses/GPL` in the Debian GNU/Linux distribution or on the World Wide Web at the GNU website. You can also obtain it by writing to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA
French version of this guide is available at http://fai-project.org/fai-guide-fr.

1 Introduction

1.1 Availability

Homepage
https://fai-project.org

FAI wiki
https://wiki.fai-project.org

Download
https://fai-project.org/download

Entry for sources.list
deb https://fai-project.org/download buster koeln

Manual pages
https://fai-project.org/doc/man/

Mailing list
https://lists.uni-koeln.de/mailman/listinfo/linux-fai

Feedback
Send feedback and comments to fai@fai-project.org or to the mailing list.

Bugs
Use the Debian bug tracking system (BTS) https://bugs.debian.org

User visible changes
https://fai-project.org/NEWS

Source tree via git
git clone git://github.com/faiproject/fai.git

View source tree via https
https://github.com/faiproject/fai

The man pages always include up-to-date information and a lot of details of all FAI commands. So, don’t forget to read them carefully. Now read this manual, then enjoy the fully automatic installation and your saved time.

1.2 Motivation

Have you ever performed identical installations of an operating system several times? Would you like to be able to install a Linux cluster with dozens of nodes single handedly?

Repeating the same task again and again is boring — and will surely lead to errors. Also a whole lot of time could be saved if the installations were done automatically. An installation process with manual interaction does not scale. But clusters have the habit of growing over the years. Think long-term rather than planning just a few months into the future.

In 1999, I had to perform an installation of a Linux cluster with one server and 16 clients. Since I had much experience doing automatic installations of Solaris operating systems on SUN SPARC hardware, the idea to build an automatic installation for Debian was born. Solaris has an automatic installation feature called JumpStart ¹. In conjunction with the auto-install scripts from Casper Dik ², I could save a lot of time not only for every new SUN computer, but also for re-installation of existing workstations. For example, I had to build a temporary LAN with four SUN workstations for a conference, which lasted only a

²http://www.science.uva.nl/pub/solaris/auto-install
few days. I took these workstations out of our normal research network and set up a new installation for the conference. When it was over, I simply integrated the workstations back into the research network, rebooted just once, and after half an hour, everything was up and running as before. The configuration of all workstations was exactly the same as before the conference, because everything was performed by the same installation process. I also used the automatic installation for reinstalling a workstation after a damaged hard disk had been replaced. It took two weeks until I received the new hard disk but only a few minutes after the new disk was installed, the workstation was running as before. And this is why I choose to adapt this technique to a PC cluster running Linux.

1.3 How does FAI work

The install client which will be installed using FAI, is booted via network card or from CD or USB stick. It gets an IP address and boots a Linux kernel which mounts its root file system via NFS (the nfsroot) from the install server. After the kernel is started, the FAI startup script performs the automatic installation which doesn’t need any interaction. First, the hard disks will be partitioned, file systems are created and then software packages are installed. After that, the new installed operating system is configured to your local needs using some scripts. Finally, the new operating system will be booted from the local disk.

The details of how to install the computer (the configuration) are stored in the configuration space on the install server. Configuration files are shared among groups of computers if they are similar using the class concept. So you need not create a configuration for every new host. Hence, FAI is a scalable method to install a big cluster with a great number of nodes even if their configuration is not identical.

FAI can also be used as a rescue system or for hardware inventory. You can boot your computer, but it will not perform an installation. Instead it will run a fully functional Debian GNU/Linux without using the local hard disks. Then you can do a remote login and backup or restore a disk partition, check a file system, inspect the hardware or do any other task.

1.4 Features

• A fully automated installation can be performed.
• Very quick unattended installation.
• Flexible system through easy class concept.
• Update of running systems without re-installation.
• Cloud images for virtualization environment.
• Hosts can boot from network card, CD, USB stick.
• Simple creation of an installation CD or USB stick.
• PXE with DHCP boot method is supported.
• ReiserFS, ext3/ext4, btrfs and XFS file system support.
• Software RAID and LVM support.
• Automatic hardware detection.
• You can deploy Debian, Ubuntu, CentOS, SuSE, Scientific Linux.
• Remote login via ssh during installation process possible.
• All similar configurations are shared among all install clients.
• Log files for all installations are saved to the installation server.
• Shell, Perl, Python, Ruby, expect and CFEngine scripts are supported during the customization step.
• Support for many protocols like NFS, FTP, HTTPS, git.
• Can be used as a rescue system and for hardware inventory.
• Diskless client support.
• Cross-architecture support e.g. for embedded devices.
• Easily add your own functions via hooks or change the default behavior.
• Cloning machines using disk images is supported.

1.5 Installation times

The installation time is determined by the amount of software and the speed of the hard disk. Here are some sample times. All install clients had a 1Gbit network card installed.

<table>
<thead>
<tr>
<th>CPU</th>
<th>RAM</th>
<th>Disk</th>
<th>Software installed</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>i7-3770T 2.50GHz</td>
<td>8GB</td>
<td>SSD</td>
<td>6 GB software</td>
<td>8.5 min</td>
</tr>
<tr>
<td>Core-i7 3.2GHz</td>
<td>6GB</td>
<td>SATA disk</td>
<td>4.3GB software</td>
<td>7 min</td>
</tr>
<tr>
<td>Core-i7 3.2GHz</td>
<td>6GB</td>
<td>SATA disk</td>
<td>471 MB software</td>
<td>77 sec</td>
</tr>
<tr>
<td>Intel Core2 Duo</td>
<td>2GB</td>
<td>SATA disk</td>
<td>3 GB software</td>
<td>14 min</td>
</tr>
</tbody>
</table>

2 Quickstart - For the impatient user

2.1 My first installation

Without further ado, this section will provide a quick and easy demonstration of a fully automatic installation using the FAI CD and a virtual machine.

Just download the CD ISO image from https://fai-project.org/fai-cd and boot your VM using this CD. You will see a grub menu where you can select from different installation types.

This installation will run without an install server. The CD installation is the same as when run in a network environment using the FAI install server and can also be used from USB stick on a real computer.

You can also create yourself a custom fully automated installation image on the webpage https://fai-project.org/FAIme without the need of installing FAI on your computer.

2.2 My first server installation

We will create a private network and start two virtual machines. One will become your own FAI server, the other will be an install client.

If you intend to use VMware or VirtualBox, ensure that your client uses a bridged network connection. A detailed description is in the FAI wiki. Also, it is not possible to use bridged network interfaces over wireless, as most WiFi network cards do not support this feature.

When using Qemu/KVM and the fai-kvm wrapper you can create the network using the command fai-mk-network. First install some packages

```
# apt-get install qemu-system-gui qemu-kvm qemu-utils bridge-utils
```

The next command sets up a private network with a software bridge with several tap devices that belong to the user <username>.

```
fai-mk-network <username>
```

After that, you can use fai-kvm (-h will give you some help) for starting virtual machines using KVM that are connected to this private network. Be careful. By default, fai-kvm will create the disk images for the virtual machines in /tmp, which is a RAM disk on most systems. It’s no problem to create an empty 20G disk image in /tmp (even if this partition is of 4GB size), but while the VM is writing data to its disk, this will start to consume space in /tmp.

Start the first virtual host, which will become the FAI server:

```
fai-kvm -Vn -s20 -u 1 cd faicd64-small.iso
```

In the grub menu select faiserver, using internal DHCP and a fixed IP. This will install a host called faiserver with IP 192.168.33.250 which contains all software needed for a FAI server. It will also set up a local package cache (using apt-cacher-ng). Once the installation is finished, reboot the machine. During the first boot of the new system, it will automatically set up the nfsroot. This may take some minutes.

After that you can start additional hosts using network boot. For every new host, you have to use a different value for -u, which will be used for generating different MAC addresses and using different disk image file names.

```
fai-kvm -Vn -u 2 pxe
```

Those install clients will show you a menu, where you can select which type of installation you like to perform. If the install client does not find the server, it is usually because fai-monitor is no longer running on it. This can happen, if you reboot the faiserver after the installation. To remedy this, simply run fai-monitor on the faiserver and re-attempt the client boot.

Another client could be started with:

```
fai-kvm -Vn -u 3 pxe
```

You can start as many machines in the network as tap devices are available. All these machines can connect to the outside internet but are only reachable from your host machine.

## 3 Overview and Concepts

FAI is a non-interactive system to install, customize and manage Linux systems and software configurations on computers as well as virtual machines and chroot environments, from small networks to large infrastructures and clusters. You can take one or more virgin PCs, turn on the power and after a few minutes Linux is installed, configured and running on the whole cluster, without any interaction necessary. Thus, it’s a scalable method for installing and updating a cluster unattended with little effort involved. FAI uses the Linux operating system and a collection of shell and Perl scripts for the installation process. Changes to the configuration files of the operating system can be made by CFEngine, shell (bash and zsh), Perl, Python, Ruby and expect scripts.

FAI’s target group are system administrators who have to install Linux onto one or even hundreds of computers. Because it’s a general purpose installation tool, it can be used for installing a Beowulf cluster, a rendering farm or a Linux laboratory or a classroom. Also large-scale Linux networks with different hardware or different installation requirements are easy to establish using FAI. But don’t forget to plan your installation. Chapter [plan] has some useful hints for this topic.

### 3.1 Important Terms

First, some terms used in this manual are described.

**install server**

It provides DHCP, TFTP and NFS services and the configuration data for all install clients. In the examples of this manual this host is called faiserver. The host where the package fai-server is installed.

4This installation will consume about 2GB of space in /tmp.
install client
A host which will be installed using FAI and a configuration provided by the install server. Also called client for short. In this manual, the example hosts are called demohost, xfcehost, gnomehost… This computer should boot from its network interface using PXE.

configuration space
A subdirectory structure containing several files. Those files describe the details of how the installation of the clients will be performed. All configuration data is stored here. It’s also called config space for short. It includes information about:

- Hard disk layout in a format similar to fstab
- Local file systems, their types, mount points and mount options
- Software packages
- Keyboard layout, time zone, Xorg configuration, remote file systems, user accounts, printers…

The package fai-doc includes a sample configuration space including examples for hosts using the XFCE and GNOME environment amongst other examples. Calling fai-mk-configspace copies these examples to /srv/fai/config. It’s recommended to study those config files and scripts for easier understanding how FAI works.

nfsroot, NFS-Root
A file system located on the install server. During the installation process it’s the complete file system for the install clients. All clients share the same nfsroot, which they mount read only. The nfsroot needs about 780MB of free disk space.

TFTP
Serves clients the initrd and kernel that is used for the installation process. Along with the file system served by NFS, these two make up a temporary OS in which the installations are performed.

FAI classes
Classes are names which determine which configuration file is selected. If a client belongs to class WEBSERVER, it will be configured as a webserver, the class DESKTOP for e.g. determines which software packages will be installed.

profile
A FAI profile is just a list of FAI classes assigned to a profile name, which is extended by a description of this profile. I.e. one could have two "Webserver" profiles, one including the APACHE class another including the NGINX class, to then install the respective webserver solution.

tasks
The installation of a client consists of several parts, which are called tasks. Tasks are predefined subroutines which perform a certain part of the FAI. The following FAI tasks are performed during an installation on the install clients.

```
confdir    # get the config space
setup      # some initialization, start sshd on demand
defclass   # define FAI classes
defvar     # define variables
action     # evaluate FAI_ACTION
install    # Start the installation
partition  # partition the harddisks, create file systems
mountdisks # mount the file systems
extrbase   # extract the base.tar.xz
debconf    # do the Debian debconf preseeding
repository # prepare access to the package repository
updatebase # Set up package tools and update packages
instsoft   # install software packages
configure  # call customization scripts
finish     # do some cleanup, show installation statistics
tests      # call tests if defined
chboot     # call fai-chboot on the install server
savelog    # save log files to local and remote location
faiend     # reboot host, eject CD if needed
```
These are tasks, which are only executed when a different action is performed

- dirinstall # install a chroot environment
- softupdate # only do the system configuration
- sysinfo # print detailed system information
- inventory # print short hardware inventory list

For a more in-depth description of tasks, see [tasks].

Note that you are not limited to the FAI tasks. You can also define additional programs or scripts which will be run on particular occasions. They are called hooks.

**hooks**

Hooks are plugins, they can add additional functionality to the installation process or even replace entire tasks of FAI. Hooks are explained in detail in [hooks].

### 3.2 The class concept

Classes are used in nearly all tasks of the installation. Classes determine which configuration files to choose from a list of available alternatives. To determine which config files to use, FAI searches the list of defined classes and uses all configuration files that match a class name[^5]. The following loop implements this function in pseudo shell code:

```sh
for class in $all_classes; do
    if [ -r $config_dir/$class ]; then # if a file with name $class exists
        your_command $config_dir/$class # call a command with this file name
        fi # exit if only the first matching file is needed
    done
```

The very nice feature of this is that you can add a new configuration alternative and it will automatically be used by FAI without changing the code, if the configuration file uses a class name.

This is because the loop automatically detects new configuration files that should be used. The idea of using classes in general and using certain files matching a class name for a configuration is adopted from the installation scripts by Casper Dik for Solaris. This technique proved to be very useful and easy.

You can group multiple hosts that share the same configuration files by using the same class. You can also split the whole configuration data for all clients into several classes and use them like lego bricks and build the entire configuration for a single client by assembling the bricks together.

If a client belongs to class A, we say the class A is defined for this client. A class has no value, it is just defined or undefined.

Classes determine how the installation is performed. For example, an install client can be configured to get the XFCE desktop by just adding the class `XFCE` to it. Naturally, also more granular configurations are possible. For instance, classes can describe how the hard disk should be partitioned, they can define which software packages will be installed, or which customization steps are performed.

Often, a client configuration is created by only changing or appending the classes to which this client belongs, making the installation of a new client very easy. Thus no additional information needs to be added to the configuration space if the existing classes suffice for your needs.

As you can see, classes are a central pillar of customizing your configuration space and with that your client installation. On how to define your own classes, refer to [defining classes].

[^5]: It's also possible to use only the configuration file with the highest priority since the order of classes define a priority from low to high within the list of classes.
4 Setup your faiserver

Here’s how to set up the install server in a few minutes. Following steps are needed:

1. Set up the install server
   a. Install FAI packages
   b. Create the nfsroot
   c. Copy the examples to the config space
   d. Configure network daemons
   e. Create the PXELINUX configurations

2. Boot and install clients

4.1 Install the FAI packages

- Install the key of the FAI project package repository:
- Add the URL of the package repository of the FAI project.
- Install the package fai-quickstart on your install server.

```
# wget -O fai-project.gpg https://fai-project.org/download/2BF8D9FE074BCDE4.gpg
# cp fai-project.gpg /etc/apt/trusted.gpg.d/
# echo "deb http://fai-project.org/download buster koeln" > /etc/apt/sources.list.d/fai.
# apt-get update
# aptitude install fai-quickstart
```

This will also install the packages for DHCP, TFTP and NFS server daemons.

4.2 Create the nfsroot

- Also enable the package repository of the FAI project in a different sources.list file which is used when building the nfsroot. Then, enable the log user for FAI.

```
# sed -i -e 's/^#deb/deb/' /etc/fai/apt/sources.list
# sed -i -e 's/#LOGUSER/LOGUSER/' /etc/fai/fai.conf
```

- By default, FAI uses http://deb.debian.org as package mirror, which should attempt to find a fast package repository for you. Now, we can run fai-setup(8) and check if everything went well. The log file is written to /var/log/fai/fai-setup.log.

```
# fai-setup -v
```

- These are some of the lines you will see at the end of fai-setup. A complete example of fai-setup.log is available on the FAI web page at https://fai-project.org/logs/fai-setup.log.

---

6 If you want to use a faster mirror, adjust the URL in /etc/fai/apt/sources.list and FAI_DEBOOTSTRAP in /etc/fai/nfsroot.conf before calling fai-setup.
7 This will call fai-make-nfsroot(8) internally.
FAI Guide (Fully Automatic Installation)

FAI packages and related packages inside the nfsroot:
dracut 050+65-1
dracut-live 050+65-1
dracut-network 050+65-1
dracut-squash 050+65-1
fai-client 5.9.4
fai-nfsroot 5.9.4
fai-setup-storage 5.9.4
Waiting for background jobs to finish
fai-make-nfsroot finished properly.
Log file written to /var/log/fai/fai-make-nfsroot.log
Adding line to /etc/exports: /srv/fai/config 192.168.33.250/25(async,ro,no_subtree_check)
Adding line to /etc/exports: /srv/fai/nfsroot 192.168.33.250/25(async,ro,no_subtree_check, ← no_root_squash)
Reloading nfs-kernel-server configuration (via systemctl): nfs-kernel-server.service.
Your initial config space is now located in /srv/fai/config
Please don't forget to fill out the FAI questionnaire after you've finished your project ← with FAI.
FAI setup finished.
Log file written to /var/log/fai/fai-setup.log

• fai-setup has created the LOGUSER, the nfsroot and has added additional lines to /etc/exports. The subdirectories added to /etc/exports are exported via NFS v3, so all install clients in the same subnet can mount them via NFS.

4.3 Creating the configuration space

Install the simple examples into the configuration space.

$ fai-mk-configspace

These examples contain configuration for some sample hosts. Depending on the host name used, your computer will be configured as follows:

demohost
A machine which needs only a small hard disk. This machine is configured with network as DHCP client, and an account demo is created.

xfcehost
A XFCE desktop is installed, using LVM, and the account demo is created.

gnomehost
A GNOME desktop is installed, and the account demo is created.

ubuntuhost
A Ubuntu desktop will be installed, and the account demo is created.

other host names
Hosts with another host name will most notably use the classes FAIBASE, DHCPC and GRUB.

All hosts will have an account called demo with password fai. The root account also has the password fai.

If the FAI flag menu is added, instead of using the host name for determining the type of installation, a menu is presented, and the user can choose a profile for the installation.

8These files need not belong to the root account.
4.4 Configure the network daemons

For booting the install client via PXE, the install server needs a DHCP and a TFTP daemon running. The package fai-quickstart has already installed the software packages for those daemons. Additionally the package of the NFS server for exporting the nfsroot and the config space was installed.

4.4.1 Configuration of the DHCP daemon

Ideally, your faiserver should also be your DHCP server. If that is not the case, instruct the admin responsible of the DHCP server to configure it according to this section. Optionally, it is possible to avoid that by using the [autodiscover] feature released in FAI 5.0.

An example for dhcpd.conf(5) is provided with the fai-doc package. Start using this example and look at all options used therein.

```bash
# cp /usr/share/doc/fai-doc/examples/etc/dhcpd.conf /etc/dhcp/
```

The only FAI specific information inside this configuration file is to set filename to fai/pxelinux.0 and to set next-server and server-name to the name of your install server. All other information is only network related data, which is used in almost all DHCP configurations. Adjust these network parameters to your local needs.

```conf
deny unknown-clients;
option dhcp-max-message-size 2048;
use-host-decl-names on;

subnet 192.168.33.0 netmask 255.255.255.0 {
  option routers 192.168.33.250;
  option domain-name "my.example";
  option domain-name-servers 192.168.33.250;
  option ntp-servers faiserver;
  option time-servers faiserver;
  server-name faiserver;
  next-server faiserver;
  filename "fai/pxelinux.0";
}
```

If you make any changes to the DHCP configuration, you must restart the daemon.

```bash
# invoke-rc.d isc-dhcp-server restart
```

If you have multiple network interfaces, you can define on which interface the server will listen in /etc/default/isc-dhcp-server. By default, the DHCP daemon writes its log messages to /var/log/daemon.log.

4.4.2 Adding a host entry to DHCP

The MAC address is given by the hardware of the network card. For each install client you collect its MAC address and to map it to an IP address and to a host name. First, we add the IP address and the hostname to /etc/hosts.

```
192.168.33.100 demohost
```

You may also add this into your Domain Name System (DNS)
The mapping from the MAC address to the IP address is done in the `dhcpd.conf` file. Here, we add a host entry using the command `dhcp-edit(8)`. Here you have to replace `01:02:03:AB:CD:EF` with the MAC you have found.

```bash
# dhcp-edit demohost 01:02:03:AB:CD:EF
```

After calling this command, this is what the host entry in `dhcpd.conf` will look like:

```plaintext
host demohost {hardware ethernet 01:02:03:AB:CD:EF;fixed-address demohost;}
```

### 4.4.3 TFTP

Normally, you do not need any changes to the TFTP daemon configuration. The files which are provided by TFTP are located in `/srv/tftp/fai`.

### 4.4.4 NFS

The command `fai-setup` has already set up the NFS daemon and added some lines to the configuration file `/etc/exports`. It exports the directories using NFS v3.

### 4.5 Creating the PXELINUX configuration

The last step before booting your client for the first time is to specify what configuration the client should boot when doing PXE boot. We use the command `fai-chboot(8)` to create a pxelinux configuration for each install client. This includes information about the kernel, the initrd, the config space and some boot parameters. You should read the manual page, which gives you some good examples. Here’s the command for starting the installation for the host `demohost`.

```bash
$ fai-chboot -IFv -u nfs://faiserver/srv/fai/config demohost
Booting kernel vmlinuz-4.19.0-5-amd64
  append initrd=initrd.img-4.19.0-5-amd64 ip=dhcp
  FAI_FLAGS=verbose,sshd,createvt FAI_CONFIG_SRC=nfs://faiserver/srv/fai/config
demohost has 192.168.33.100 in hex C0A82164
Writing file /srv/tftp/fai/pxelinux.cfg/C0A82164 for demohost
```

At this point, you should have a working faiserver setup and your clients should boot into FAI and be able to install one of the examples.

In the following section, you can read about planning your installation, tailoring your configuration space to your particular needs and extending FAI using hooks.

### 4.6 Custom server

The faiserver and its setup is by no means static. It is possible to customize and extend your server. For this, please refer to the [Customizing your install server setup] section in [advanced].
5 Plan your installation

Before starting your installation, you should invest a lot of time into planning your installation. Once you’re happy with your installation concept, FAI can do all the boring and repetitive tasks to turn your plans into reality. FAI can’t do good installations if your concept is imperfect or lacks some important details. Start planning the installation by answering the following questions:

- Will I create a Beowulf cluster, or do I have to install some desktop machines?
- What does my LAN topology look like?
- Do I have uniform hardware? Will the hardware stay uniform in the future?
- Does the hardware need a special kernel?
- How should the hosts be named?
- How should the local hard disks be partitioned?
- Which applications will be run by the users?
- What software should be installed?
- Which daemons should be started, and what should the configuration for these look like?
- Which remote file systems should be mounted?
- How should backups be performed?

You also have to think about user accounts, printers, a mail system, cron jobs, graphic cards, dual boot, NIS, NTP, timezone, keyboard layout, exporting and mounting directories via NFS and many other things. So, there’s a lot to do before starting an installation. And remember that knowledge is power, and it’s up to you to use it. Installation and administration is a process, not a product. FAI can’t do things you don’t tell it to do.

You don’t need to start from scratch. Look at the files and scripts in the configuration space. There are a lot of things you can use for your own installation. A good paper called "Bootstrapping an Infrastructure" with more aspects of building an infrastructure is available at http://www.infrastructures.org/papers/bootstrap/bootstrap.html

5.1 The configuration space and its subdirectories

The configuration space is the collection of information about how exactly to install a client. The central configuration space for all install clients is located on the install server in /srv/fai/config and its subdirectories. This will be mounted by the install clients to /var/lib/fai/config. The main installation command fai(8) uses all these subdirectories in the order listed except for hooks.

**class/**  
Scripts and files to define classes and variables.

**disk_config/**  
Configuration files for disk partitioning, software RAID, LVM and file system creation.

**basefiles/**  
Normally the file base.tar.xz (located inside the nfsroot) is extracted on the install client after the new file systems are created and before package are installed. This is a minimal base image, created right after calling debootstrap during the creation of the nfsroot on the install server. If you want to install another distribution than the nfsroot is, you can put a tar file into the subdirectory basefiles/ and name it after a class. Then the command ftar(8) is used to extract the tar file based on the classes defined. Thus the file has to be named CLASS.tar.xz not CLASS.base.tar.xz. This is done in task extrbase. Use this if you want to install another distribution or version than that running during the installation.

This basefile can also be received based on FAI classes via HTTP/HTTPS or FTP by defining the variable FAI_BASEFILEURL. FAI will download a file CLASSNAME.tar.xz (or tgz, or tar.gz,...) from this URL, if CLASSNAME matches a FAI class.

Example:
The folder must support directory listing. FAI will not probe for potentially matching files.
See chapter [otherdists] for how to install different distributions.

**debconf/**
This directory holds all *debconf*(7) data. The format is the same that is used by *debconf-set-selections*(1).

**package_config/**
Files with class names contain lists of software packages to be installed or removed by *install_packages*(8). Files named `<CLASS>.gpg` are added to the list of keys used by apt for trusted package repositories.

**scripts/**
Scripts for your local site customization. Used by *fai-do-scripts*(1).

**files/**
Files used by customization scripts. Most files are located in a subtree structure which reflects the ordinary directory tree. For example, the templates for *nsswitch.conf* are located in `$FAI/files/etc/nsswitch.conf` and are named according to the classes that they should match: `$FAI/files/etc/nsswitch.conf/NIS` is the version of `/etc/nsswitch.conf` to use for the NIS class. Note that the contents of the files directory are not automatically copied to the target machine, rather they must be explicitly copied by customization scripts using the *fcopy*(8) command.

**hooks/**
Hooks are user defined programs or scripts, which are called during the installation process. They can extend or replace the default tasks. The file name must be of format `taskname.CLASSNAME[.sh]`. A hook called `updatebase.DEBIAN` is executed prior to the task `updatebase` and only if the install client belongs to the class DEBIAN.

### 5.2 Defining classes

There are different possibilities to define classes:

1. Some default classes are defined for every host: `DEFAULT`, `LAST` and its host name.
2. Classes may be listed within a file (by default in `class/<hostname>`).
3. Classes may be dynamically defined by scripts.

The last option is a very nice feature, since these scripts will define classes in a very flexible way. For example, several classes may be defined only if certain hardware is identified or a class is defined depending on the network subnet information.

All names of classes, except the host name, are written in uppercase. They must not contain a hyphen, a hash, a semicolon or a dot, but may contain underscores and digits.

The task *defclass* calls the command *fai-class*(1) to define classes. All scripts matching `^[0-9][0-9]*` (they start with two digits) in the subdirectory `$FAI/class` are executed for defining classes. Everything that is printed to STDOUT is automatically defined as a class. For more information on defining classes, read the manual pages for *fai-class*(1). The script `50-host-classes` (see below a stripped version) is used to define classes depending on the host name.

```bash
# use a list of classes for our demo machines
case $HOSTNAME in
demohost)
  echo "FAIBASE GRUB DHCPC DEMO" ;;
xfcehost)
  echo "FAIBASE GRUB DHCPC DEMO XORG XFCE";;
faserver)
  echo "FAIBASE DEBIAN DHCPC DEMO FAISERVER" ;;
  *)
  echo "FAIBASE GRUB DHCPC" ;;
esac
```
Host names should rarely be used for the configuration files in the configuration space. Instead, a class should be defined and then added for a given host. This is because most of the time the configuration data is not specific for one host, but can be shared among several hosts.

The order of the classes is important because it defines the priority of the classes from low to high.

### 5.3 Defining variables

The task `defvar` defines the variables for the install client. Variables are defined by scripts in `class/*.var`. All global variables can be set in `DEFAULT.var`. For groups of hosts use a class file. For a single host use the file `$HOSTNAME.var`. Also here, it’s useful to study all the examples.

The following variables are used in the examples and may also be useful for your installation:

- **FAI_ACTION**
  - Set the action FAI should perform. Normally this is done by `fai-chboot(8)`. If you can’t use this command, define this variable i.e. in the script `LAST.var`.

- **FAI_ALLOW_UNSIGNED**
  - If set to 1, FAI allows the installation of packages from unsigned repositories.

- **CONSOLEFONT**
  - Is the font which is loaded during installation by `setfont(8)`.

- **KEYMAP**
  - Defines the keyboard map files in `/usr/share/keymaps` and `$FAI/files`. You don’t need to specify the full path, since this file will be located automatically.

- **ROOTPW**
  - The encrypted root password for the new system. You can use `crypt(3)`, md5 and other hash types for the password. Use `mkpasswd(1)` for creating the hash for a certain password. For example, to generate a md5 hash for the password use

  ```
  $ echo "yoursecrectpassword" | mkpasswd -m md5 -s
  ```

- **UTC**
  - Set hardware clock to UTC if `UTC=yes`. Otherwise set clock to local time. See `clock(8)` for more information.

- **TIMEZONE**
  - Is the file relative to `/usr/share/zoneinfo/` which indicates your time zone. E.g.: `TIMEZONE=Europe/Berlin`.

- **MODULESLIST**
  - A list of kernel modules which are loaded during boot of the new system (written to `/etc/modules`).

### 5.4 Hard disk configuration

The tool `setup-storage(8)` reads a file in `$FAI/disk_config` for the disk configuration. This file describes how all the local disks will be partitioned, which file system types should be created (like ext3/4, xfs, btrfs), and where they are mounted to. You can also create software RAID and LVM setups using this config file. It’s also possible to preserve the disk layout or to preserve the data on certain partitions.

During the installation process all local file systems are mounted relative to `/target`. For example if you specify the mount point `/home` in a disk configuration file this will be the directory `/target/home` during the installation process and will become `/home` for the new installed system.
5.5 Extract base file

A base file is only needed when installing a distribution which is different from the one in the nfsroot.

5.6 Debconf preseeding

You can use the format described in debconf-set-selections(1).

5.7 Access to the package repository

FAI supports http, https and NFS for accessing the package mirror. Set the variable $FAI_DEBMIRROR for using NFS.

5.8 Software package configuration

Before installing packages, FAI will add the content of all files named package_config/class.asc to the list of apt keys. If your local repository is signed by your keyid AB12CD34 you can easily add this key, so FAI will use it during installation. Use this command for creating the CLASS.asc file:

```
faiserver$ gpg -a --export AB12CD34 > /srv/fai/config/package_config/MYCLASS.asc
```

The script install_packages(8) installs the selected software packages. It reads all configuration files in $FAI/package_config whose file name matches a defined class. The syntax is very simple.

```
# an example package class

PACKAGES taskinst
  german

PACKAGES aptitude
  adduser netstd ae
  less passwd

PACKAGES remove
  gpm xdm

PACKAGES aptitude GRUB
  lilo- grub
```

Comments are starting with a hash (#) and are ending at the end of the line. Every package command begins with the word PACKAGES followed by a command name, which maps to a different package tool like apt-get, aptitude or yum for e.g. The command defines which command will be used to install the packages named after this command. The list of all available commands can be listed using install_packages -H. Supported package tools are: aptitude, apt-get, smart, yast, yum, rpm, zypper

- **hold**
  - Put a package on hold. This package will not be handled by dpkg, e.g not upgraded.

- **install**
  - Install all packages (using apt-get) that are specified in the following lines. If a hyphen is appended to the package name (with no intervening space), the package will be removed, not installed. All package names are checked for misspellings. Any package which does not exist, will be removed from the list of packages to install. So be careful not to misspell any package names.

- **install-norec**
  - Like install but without installing the recommended packages.
remove
Remove all packages that are specified in the following lines. Append a + to the package name if the package should be
installed.

taskinst
Install all packages belonging to the tasks that are specified in the following lines using tasksel(1). You can also use
aptitude for installing tasks.

aptitude
Install all packages with the command aptitude. This will be the default in the future and may replace apt-get and
taskinst. Aptitude can also install task packages.

aptitude-r
Same as aptitude with option --with-recommends.

unpack
Download package and unpack only. Do not configure the package.

dselect-upgrade
Set package selections using the following lines and install or remove the packages specified. These lines are the output
of the command dpkg --get-selections. It’s not recommended to use this format, since you are also specifying all packages
which are only installed because of a dependency or a recommends. It’s better just to specify the package you like to have,
and to let FAI (and apt-get) resolve the dependencies.

Multiple lines with lists of space separated names of packages follow the PACKAGES lines. All dependencies are resolved.
Packages with suffix - (eg. lilo-) will be removed instead of installed. The order of the packages doesn’t matter. If you
like to install packages from another release than the default, you can append the release name to the package name like in
openoffice.org/etch-backports. You can also specify a certain version like apt=0.3.1. More information on these features are
described in aptitude(8).

A line which contains the PRELOADRM commands, downloads a file using wget(1) into a directory before installing the
packages. Using the file: URL, this file is copied from $FAI_ROOT to the download directory. For example the package
realplayer needs an archive to install the software, so this archive is downloaded to the directory /root. After installing the
packages this file will be removed. If the file shouldn’t be removed, use the command PRELOAD instead.

It’s possible to append a list of class names after the command for apt-get. So this PACKAGES command will only be executed
when at least one of the corresponding classes is defined (logical OR). So you can combine many small files into the file
DEFAULT. WARNING! Use this feature only in the file DEFAULT to keep everything simple. See this file for some examples.

If you want to remove a package name from a certain class was part of this class before, you should not remove the package
name from the class file, but instead append a dash (-) to it. This will make sure that the package is removed during a softupdate
on hosts which were installed using the old class definition which included this package name.

If you specify a package that does not exist this package will be removed automatically from the installation list only if the
command install is used.

The concept of classes priority allows a higher priority class (one that comes later in the sequence of classes) to override the
selection of packages of a lower priority class. For this to work correctly, the higher priority class must use the same PACKAGES
command (e.g. PACKAGES install-norec instead of just PACKAGES install) as the one used by the class it is trying to override.
This is useful to suppress installation of a package, for example, to avoid installing the linuxlogo package installed by class
FAIBASE:

```
# example of how to override:
#
# On FAIBASE we have:
#   PACKAGES install-norec
#   linuxlogo
#
# We want to _not_ install linuxlogo, and it is in a
# install-norec section, so we must also use install-norec.
```
5.9 Customization scripts

The command `fai-do-scripts(1)` is called to execute all scripts in this directory. If a directory with a class name exists, all scripts matching `^[0-9][0-9]*` are executed in alphabetical order. So it’s possible to use scripts of different languages (shell, cfengine, Perl, Python, Ruby, expect..) for one class.

Those scripts write their output to different log files, depending on the type of script. For e.g. all shell scripts write their log to `shell.log`.

5.9.1 Shell scripts

Most scripts are Bourne shell scripts. Shell scripts are useful if the configuration task only needs to call some shell commands or create a file from scratch. In order not to write many short scripts, it’s possible to use the `ifclass` command for testing if certain classes are defined.

```
ifclass -o A B C
```

checks if one of classes A, B or C are defined. Using `-a` (logical AND) checks if all classes of a list are defined. The command `ifclass C` checks if only class C is defined.

For copying files with classes, use the command `fcopy(8)`. If you want to extract an archive using classes, use `ftar(8)`. For appending lines to a configuration file use `ainsl(1)` instead of just `echo string >> filename`.

FAI also supports `zsh(1)` scripts during the customization task. Within scripts, the variable `$classes` holds a space separated list with the names of all defined classes.

5.9.2 Cfengine scripts

CFEngine has a rich set of functions to edit existing configuration files, e.g `LocateLineMatching`, `ReplaceAll`, `InsertLine`, `AppendIfNoSuchLine`, `HashCommentLinesContaining`. But it can’t handle variables which are undefined. If a variable is undefined, the whole cfengine script will abort.

More information can be found in the manual page `cfengine(8)` or at the cfengine homepage https://www.cfengine.com.

5.10 Hooks

Hooks let you specify functions or programs which are run at certain steps of the installation process. Before a task is called, FAI searches for existing hooks for this task and executes them. As you might expect, classes are also used when calling hooks. Hooks are executed for every defined class. You only have to create the hook with the name for the desired class and it will be used. If several hooks for a task exists, they are called in the order defined by the classes. If `debug` is included in `$FAI_FLAG` the option `-d` is passed to all hooks, so you can debug your own hooks. If some default tasks should be skipped, use the subroutine `skiptask` and a list of default tasks as parameters. In the examples provided, the hooks of the class CENTOS skips some Debian specific tasks.

The directory `$FAI/hooks/` contains all hooks. A hook is an executable file following the naming scheme `taskname.CLASSNAME.[.sh]` (e.g. `repository.CENTOS` or `savelog.LAST.sh`). The task name specifies which task to precede executing this hook, if the specified class is defined for the installing client. See section [tasks] for a complete list of default tasks that can be used.

A hook of the form `hookprefix.classname` can’t define variables for the installation script, because it’s a subprocess. But you can use any binary executable or any script you wrote. Hooks that have the suffix `.sh` (e.g. `partition.DEFAULT.sh`) must be Bourne shell scripts and are sourced. So it’s possible to redefine variables for the installation scripts.
In the first part of FAI, all hooks with prefix `confdir` are called. Those hooks can not be located in the config space, since it’s not yet available. Therefore these hooks are the only hooks located in `$nfsroot/$FAI/hooks` on the install server. All other hooks are found in `$FAI_CONFIGDIR/hooks` on the install server.

All hooks that are called before classes are defined can only use the following classes: `DEFAULT $HOSTNAME LAST`. If a hook for class `DEFAULT` should only be called if no hook for class `$HOSTNAME` is available, insert these lines to the default hook:

```
hookexample.DEFAULT:

#!/bin/sh

# skip DEFAULT hook if a hook for $HOSTNAME exists
scriptname=$(basename $0 .DEFAULT)
[-f $FAI/hooks/$scriptname.$HOSTNAME ] && exit

# here follows the actions for class DEFAULT
.
```

Some examples for what hooks could be used:

- Load kernel modules before classes are defined in `$FAI/class`.
- Send an email to the administrator if the installation is finished.
- Install a diskless client and skip local disk partitioning.
- Have a look at `hooks/debconf.IMAGE` for how to clone a machine using a file system image.

### 5.11 FAI flags

The variable `$FAI_FLAGS` contains a space separated list of flags. Flags are normally defined in the pxelinux.cfg file which should be created by `fai-chboot(1)`. The following flags are known:

**verbose**

Create verbose output during installation. This should always be the first flag, so consecutive definitions of flags will be verbosely displayed.

**debug**

Create debug output. No unattended installation is performed. During package installation you have to answer all questions of the postinstall scripts on the client’s console. A lot of debug information will be printed out. This flag is only useful for FAI developers.

**sshd**

Start the ssh daemon to enable remote logins. You can then log in as `root` to all install clients during the installation. The default password is `fai` and can be changed by setting `FAI_ROOTPW` in `nfsroot.conf(5)`. To log in from your server to the install client (named demohost in this example) use:

```
$ ssh root@demohost
Warning: Permanently added 'demohost,192.168.33.100' to the list of known hosts.
root@demohost's password:
```

This is only the root password during the installation process, not for the new installed system. You can also log in without a password when using `$SSH_IDENTITY`.

**createvt**

Create two virtual terminals and execute a bash if `ctrl-c` is typed in the console terminal. The additional terminals can be accessed by typing `Alt-F2` or `Alt-F3`. Otherwise, no terminals are available and typing `ctrl-c` will reboot the install client. Setting this flag is useful for debugging. If you want an installation which should not be interruptible, do not set this flag.
menu
This enables a user menu for selecting a profile. All files `class/*.profile` are read and a curses based menu will be created.

reboot
Reboot the install client after installation is finished without typing RETURN on the console. If this flag is not set, and error.log contains anything, the install client will stop and wait that you press RETURN. If no errors occurred, the client will always reboot automatically.

halt
Halt the install client at the end of the installation, instead of rebooting into the new system.

initial
Used by `setup-storage(8)`. Partitions marked with `preserve_reinstall` are preserved unless this flag is set. Often, this flag is set in a file `class/*.var` by using setting `flag_initial=1`.

6 FAI installs your plan

6.1 The early part of an installation

After the kernel has booted, it mounts the root file system via NFS from the install server and starts the script `/usr/sbin/fai` 10. This script controls the sequence of the installation. No other scripts in `/etc/init.d/` are used.

The configuration space is made available via the configured method (an NFS mount by default) from the install server to the path defined in `$FAI` 11.

6.2 Boot messages

When booting the install client from network card with PXE you will see some messages like this:

```plaintext
Managed PC Boot Agent (MBA) v4.00
Pre-boot eXecution Environment (PXE) v2.00
DHCP MAC ADDR: 00 A2 A3 04 05 06
DHCP.../
CLIENT MAC ADDR: 00 A2 A3 04 05 06 GUIDE: 3D6C4552
CLIENT IP: 192.168.33.100 MASK: 255.255.255.0 DHCP IP: 192.168.33.250
GATEWAY IP: 192.168.33.1

!PXE entry point found (we hope) at 9854:0106 via plan A
UNDI code segment at: 9854 len 5260
UNDI data segment at: 921D len 63A2
Getting cached packet 01 02 03
My Ip address seems to be C0A82164 192.168.33.100
ip=192.168.33.100:192.168.33.250:192.168.33.1:255.255.255.0
BOOTIF=01-00-A2-A3-04-05-06
SYSUUID=
TFTP prefix: fai/
Trying to load pxelinux.cfg/C0A82164

Loading vmlinux-4.9.0-3-amd64..................
Loading initrd.img-4.9.0-3-amd64................ready.
```

10Since the root file system on the clients is mounted via NFS, `fai` is located in `/srv/fai/nfsroot/usr/sbin` on the install server.
11$FAI is an internal variable used by the FAI scripts. By default the path is `/var/lib/fai/config`.
At this point the install client has successfully received the network config via DHCP and the kernel and initrd via TFTP. It now boots the Linux kernel and the initrd. If everything went right, the initrd mounts the `nfsroot` and the FAI scripts are started.

The first thing you see is the red FAI copyright message.

---

**Fully Automatic Installation - FAI**

5.5 (c) 1999-2017

Thomas Lange <lange@informatik.uni-koeln.de>

---

Calling task_confdir

Kernel currently running: Linux 4.9.0-3-amd64 x86_64 GNU/Linux

Kernel parameters: `BOOT_IMAGE=vmlinuz-4.9.0-3-amd64 initrd=initrd.img-4.9.0-3-amd64` \ `rw rootvklp=dhcp root=192.168.33.250:/srv/fai/nfsroot FAI_FLAGS=verbose,sshd,createvt` \ `FAI_CONFIG_SRC=nfs://faiserver/srv/fai/config` FAI_ACTION=install

Reading `/tmp/fai/boot.log`

FAI_FLAGS: verbose sshd createvt

Setting SERVER=faiserver. Value extracted from FAI_CONFIG_SRC.

FAI_CONFIG_SRC is set to nfs://faiserver/srv/fai/config

Configuration space faiserver:/srv/fai/config mounted to /var/lib/fai/config

Source hook: setup.DEFAULT.sh

setup.DEFAULT.sh OK.

Calling task_setup

FAI_FLAGS: verbose sshd createvt

15 Sep 13:22:37 ntpdate[1533]: step time server 192.168.33.250 offset -0.342793 sec

Press ctrl-c to interrupt FAI and to get a shell

Starting FAI execution - 20170915_132237

Calling task_defclass

fai-class: Defining classes.

Executing `/var/lib/fai/config/class/01-short-hostname.sh`.

01-short-hostname.sh OK.

Executing `/var/lib/fai/config/class/10-base-classes`.

10-base-classes OK.

Executing `/var/lib/fai/config/class/20-hwdetect.source`.

Loading kernel module md-mod

20-hwdetect.source OK.

Executing `/var/lib/fai/config/class/50-host-classes`.

50-host-classes OK.

List of all classes: DEFAULT LINUX AMD64 FAIBASE DHCPC DEMO GRUB client01 LAST

You can also see the list of FAI classes, that are defined for this host. This list is very important for the rest of the installation.

The first task is called *confdir*, which is responsible for getting access to the config space. Here, we use an NFS mount from the install server as you can see on the console (and later in the logs).

---

### 6.3 Rebooting the computer into the new system

For rebooting the computer during or at the end of the installation you should use the command `faireboot` in favour of the normal reboot command. Use `faireboot` also if logged in from remote. If the installation hasn’t finished, use `faireboot -s`, so the log files are also copied to the install server.

\[\text{12/srv/fai/nfsroot} \text{from the install server via NFS}\]
If the installation has finished successfully, the computer should boot a small Debian system. You can login as user demo or root with password fai.

6.4 Starting FAI (task confdir)

After the install client has booted only the script /usr/sbin/fai is executed. It will do some minimal initialization. The variable $FAI_CONFIG_SRC \(^{13}\) is used to get access to the FAI configuration space which is then available in the directory $FAI \(^{14}\). FAI will not proceed without the config space.

6.5 Defining classes and variables (tasks defclass and defvar)

The command fai-class(1) executes scripts in $FAI/class for defining classes. If the scripts write a string to stdout, this will be defined as a class. Read all the details in the man page of fai-class(1).

After defining the classes, every file matching .var with a prefix which matches a defined class is sourced to define variables. It must contain valid shell code.

6.6 Partitioning local disks, creating file systems (task partition)

For the disk partitioning exactly one disk configuration file from $FAI/disk_config is selected using classes.

The format of the disk configuration is similar to a fstab file.

The partitioning tool setup-storage(8) performs all commands necessary for creating the disk partition layout, software RAID, LVM and for creating the file systems. Disks and partitions may easily be referenced by disk1.1, disk2.4 etc. Read the manual page of setup-storage(8) for a detailed description and some examples of the format.

6.7 Debconf preseeding (task debconf)

Files in $FAI/debconf are used for the usual debconf(7) preseeding if the file names match a class name.

6.8 Installing software packages (task instsoft)

The command install_packages(8) reads the config files from $FAI/package_config in a class based manner and installs software packages on the new file system.

It installs the packages using apt-get(8), aptitude(1), yum or other package tools without any manual interaction needed. Package dependencies are also resolved by the package tools.

The format of the configuration files is described in [packageconfig].

6.9 Site specific customization (task configure)

Often the default configurations of the software packages will not meet your site-specific needs. You can call arbitrary scripts which adjust the system configuration. Therefore the command fai-do-scripts(1) executes scripts in $FAI/scripts in a class based manner. It is possible to have several scripts of different types (shell, cfengine, ...) to be executed for one class.

The default set of scripts in $FAI/scripts include examples for installing Debian and CentOS machines. They set the root password, add a user account (set by $username, default to demo), set the timezone, configure the network for DHCP or using a fixed IP address, setup grub and more. They should do a reasonable job for your installation. You can edit them or add new scripts to match your local needs.

More information about these scripts are described in [cscripts].

\(^{13}\)It is defined on the kernel command line

\(^{14}\)Var/lib/fai/config
6.10 Saving log files (task savelog)

When all tasks are finished, the log files are written to `/var/log/fai/$HOSTNAME/install/` on the new system and to the account on the install server if `$LOGUSER` is defined. It is also possible to specify another host as log saving destination through the variable `$LOGSERVER`. If `$LOGSERVER` is not defined, FAI uses the variable `$SERVER` which is only defined during an initial installation (by `get-boot-info`). Make sure to set `$LOGSERVER` in a `class/*.var` script if you are using the action `softupdate`.

Additionally, two symlinks will be created to indicated the last directory written to. The symlink `last` points to the log directory of the last FAI action performed. The symlinks `last-install` and `last-sysinfo` point to the directory of the last corresponding action.

By default log files will be copied to the log server using `scp`. You can use the variable `$FAI_LOGPROTO` in file `fai.conf(5)` to choose another method for saving logs to the remote server. Here’s an example of the symlink structure:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Owner</th>
<th>Group</th>
<th>Link</th>
<th>Date</th>
<th>Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>lrwxrwxrwx</td>
<td>1 fai fai 23 Dec 2</td>
<td>2013</td>
<td>last-sysinfo</td>
<td>-&gt;</td>
<td>sysinfo-20131202_161237</td>
<td></td>
</tr>
<tr>
<td>drwxr-xr-x</td>
<td>2 fai fai 4096 Dec 2</td>
<td>2013</td>
<td>sysinfo-20131202_161237</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drwxr-xr-x</td>
<td>2 fai fai 4096 Feb 14</td>
<td>2014</td>
<td>install-20140214_142150</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drwxr-xr-x</td>
<td>2 fai fai 4096 Dec 2</td>
<td>11:47</td>
<td>install-20141202_113918</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lrwxrwxrwx</td>
<td>1 fai fai 23 Dec 4</td>
<td>13:22</td>
<td>last-install</td>
<td>-&gt;</td>
<td>install-20141204_131351</td>
<td></td>
</tr>
<tr>
<td>lrwxrwxrwx</td>
<td>1 fai fai 23 Dec 4</td>
<td>13:22</td>
<td>last-sysinfo</td>
<td>-&gt;</td>
<td>install-20141204_131351</td>
<td></td>
</tr>
<tr>
<td>drwxr-xr-x</td>
<td>2 fai fai 4096 Dec 4</td>
<td>13:22</td>
<td>install-20141204_131351</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples of the log files can be found at [https://fai-project.org/logs](https://fai-project.org/logs).

6.11 Reboot the new installed system

Before rebooting, the install client calls `fai-chboot -d <hostname>` on the install server, to disable its own PXELINUX configuration. Otherwise, it would restart the installation during the next boot. Normally this should boot the new installed system from its second boot device, the local hard disk.

At the end, the system is automatically rebooted if "reboot" was added to `$FAI_FLAGS`.

7 Advanced FAI topics

7.1 Checking parameters received from DHCP servers

If the install client boots you can check if all information from the DHCP daemon are received correctly. The received information is written to `/tmp/fai/boot.log`. An example of the result of a DHCP request can be found in the sample log files.

7.2 Monitoring multiple client installations

You can monitor the installation of all install clients with the command `fai-monitor(8)`. All clients check if this daemon is running on the install server (or the machine defined by the variable `$monserver`). Each time a task starts or ends, a message is sent. The FAI monitor daemon prints this messages to standard output. There’s also a graphical frontend available, called `fai-monitor-gui(1)`.

```
$ fai-monitor | fai-monitor-gui -&
```
7.3 Collecting Ethernet addresses for multiple hosts

You have to collect all Ethernet (MAC) addresses of the install clients and assign a host name and IP address to each client. To collect the MAC addresses, boot your install clients. You can already do this before any DHCP daemon is running in your subnet. They will fail to boot (because of the missing DHCP or missing TFTP), but you can still collect the MAC addresses.

While the install clients are booting, they send broadcast packets to the LAN. You can log the MAC addresses of these hosts by running the following command simultaneously on the server:

```
faiserver# tcpdump -qtel broadcast and port bootpc >/tmp/mac.list
```

After the hosts have been sent some broadcast packets abort tcpdump by typing `ctrl-c`. You get a list of all unique MAC addresses with these commands:

```
faiserver$ perl -ane 'print "\U$F[0]\n"' /tmp/mac.list|sort|uniq
```

After that, you only have to assign these MAC addresses to host names and IP addresses (/etc/ethers and /etc/hosts or corresponding NIS maps). With this information you can configure your DHCP daemon (see the section [bootdhcp]).

7.3.1 Debugging the network traffic

If the client can’t successfully boot from the network card, use `tcpdump(8)` to look for Ethernet packets between the install server and the client. Search also for entries in several log files made by `tftpd(8)` and `dhcpcd(8)`:

```
faiserver$ egrep "tftpd|dhcpcd" /var/log/*
```

7.4 Details of PXE booting

Here we describe the details of PXE booting, which are only needed if you have problems when booting your install clients. Almost all modern bootable network cards support the PXE boot environment. PXE is the Preboot Execution Environment. This requires the PXELINUX bootloader and a special version of the TFTP daemon, which is available in the Debian packages `pxelinux` and `tftpd-hpa`. PXE booting also needs a DHCP server, so that the network card can configure its IP parameters.

This is the sequence of a PXE boot:

1. Network card of the client sends its MAC address
2. DHCP server replies with IP configuration for the client
3. Network card configures IP
4. Install client gets the pxelinux.0 binary via TFTP
5. Get the pxelinux.cfg/C0A8210C configuration file via TFTP
6. C0A8210C is the IP address of the client in hexadecimal
7. This configuration contains kernel, initrd and additional kernel command line parameters, which was created by `fai-chboot`
8. Get the kernel and initrd via TFTP

I recommend to write the MAC addresses (last three bytes will suffice if you have network cards from the same vendor) and the host name in the front of each chassis.

---
Example of a pxelinux.cfg file:

```
default fai-generated
label fai-generated
kernel vmlinuz-4.19.0-5-amd64
append initrd=initrd.img-4.19.0-5-amd64 ip=dhcp root=/srv/fai/nfsroot:vers=3 rootvol
FAI_FLAGS=verbose,sshd,createvt FAI_CONFIG_SRC=nfs://faiserver/srv/fai/config FAI_ACTION=install
```

See `/usr/share/doc/syslinux/pxelinux.doc` for more detailed information about PXELINUX. FAI uses the `lpixelinux.0` binary which also supports loading the kernel and initrd via FTP or HTTP. The command `fai-chboot(8)` supports this with the option `-U`.

### 7.5 Customizing your install server setup

- local/faster package mirror
- different loguser
- local root pw inside nfsroot

The configuration for the FAI package (not the configuration data for the install clients) is defined in `fai.conf(5)`. Definitions that are only used for creating the nfsroot are located in `nfsroot.conf(5)`. Check these important variables in `nfsroot.conf` before calling `fai-setup` or `fai-make-nfsroot`.

**FAI_DEBOOTSTRAP**

Building the nfsroot uses the command `debootstrap(8)`. It needs the location of a Debian mirror and the name of the distribution (wheezy, jessie, stretch, sid) for which the basic Debian system should be built. Do not use different distributions here and in `/etc/fai/apt/sources.list`. This will create a broken nfsroot.

**NFSROOT_ETC_HOSTS**

This variable is only needed if the clients do not have access to a DNS server. This multiline variable is added to `/etc/hosts` inside the nfsroot. Then the install clients can access those hosts by name without using DNS.

The content of `/etc/fai/apt/sources.list` is used by the install server and also by the clients. If your install server has multiple network cards and different host names for each card (as for a Beowulf server), use the install server name which is known by the install clients.

If you have problems running `fai-setup`, they usually stem from `fai-make-nfsroot(8)` which is called by former command. Adding `-v` gives you a more verbose output which helps you pinpoint the error. The output is written to `/var/log/fai/fai-make-nfsroot.log`. 17

The setup also creates the account `fai` (defined by `$LOGUSER`) if not already available. So you can add a user before calling `fai-setup(8)` using the command `adduser(8)` and use this as your local account for saving log files. The log files of all install clients are saved to the home directory of this account. You should change the primary group of this account, so this account has write permissions to `/srv/tftp/fai` in order to call `fai-chboot` for creating the PXE configuration for the hosts.

When you make changes to `fai.conf`, `nfsroot.conf` the nfsroot has to be rebuilt by calling `fai-make-nfsroot(8)`. If you only like to install a new kernel package to the nfsroot add the flags `-k` or `-K` to `fai-make-nfsroot`. This will not recreate your nfsroot, but only updates your kernel and kernel modules inside the nfsroot or add additional packages into the nfsroot.

17For debugging purpose it may help to enter the chroot environment manually using this command. `faiserver# chroot /srv/fai/nfsroot bash`
7.6 Creating a FAI CD or and USB stick

You can easily create an installation CD (or USB stick) of your network installation setup. This will perform the same installation and configuration from CD without the need of the install server. Therefore you need to create a partial mirror of all Debian packages needed for your FAI classes (using fai-mirror(1)). Then the command fai-cd(8) will put this mirror, the nfsroot and the config space onto a bootable CD. That’s it!

To easily create the installation CD, you can use the following command (for more tuning details see fai-cd(8)):

```
faiserver# fai-cd -m <partialMirrorDir> fai-cd.iso
```

This installation CD contains all data needed for the installation. The command fai-cd(8) puts the nfsroot, the configuration space and a subset of the Debian mirror onto a CD-ROM. A partial package mirror is created using the command fai-mirror(1) which contains all packages that are used by the classes used in your configuration space. A sample ISO image is available at https://fai-project.org/fai-cd.

Using the command dd(1) you can also create a bootable USB stick by just writing the content of the ISO file to your USB stick (here the stick is /dev/sdf).

```
faiserver# dd if=fai-cd.iso of=/dev/sdf bs=1M
```

This is no live CD of the install server.

7.7 Creating VM disk images using FAI

The command fai-diskimage(8) creates disk images, which can be used with a virtual machine like KVM, VMware, VirtualBox or a cloud service like OpenStack, GCE, EC2 and others. The installation process performs the normal FAI tasks on a raw disk image. After the installation you can boot the disk image and have a running system. The disk image can also be converted to qcow2 format. You do not need to setup the FAI nfsroot when only using fai-diskimage. But you need a basefile in your configuration space. Setting the variable $FAI_BASEFILEURL will automatically download an appropriate base file into your config space.

Here’s an example how to create a raw disk image for a host called cloud3, with a small set of software packages:

```
# export FAI_BASEFILEURL=https://fai-project.org/download/basefiles/
# fai-diskimage -vu cloud3 -S2G -cDEBIAN,BUSTER64,AMD64,FAIBASE,GRUB_PC,DHCPC,DEMO,CLOUD, ←
# LAST disk.raw
```

This command will create a disk image called ubuntu.qcow2 for a Ubuntu 16.04 desktop with hostname set to foobar.

```
# export FAI_BASEFILEURL=https://fai-project.org/download/basefiles/
# cl=DHCPC,UBUNTU,XENIAL,XENIAL64,AMD64,XORG,LAST
# fai-diskimage -Nvu foobar -S5G -c$cl ubuntu.qcow2
```

You can give disk images a try without installing FAI, if you visit https://fai-project.org/FAIme/cloud

7.8 Building cross-architecture disk images

Starting FAI 5.4 it’s now possible to build a disk image for different architectures than the host is running. For example you can build an image for 64-bit ARM architecture (aarch64) on a host running on amd64 architecture. Here are the steps do to:

```
# export FAI_BASEFILEURL=https://fai-project.org/download/basefiles/
# cl=DHCPC,UBUNTU,XENIAL,XENIAL64,AMD64,XORG,LAST
# fai-diskimage -Nvu foobar -S5G -c$cl ubuntu.qcow2
```

You can give disk images a try without installing FAI, if you visit https://fai-project.org/FAIme/cloud
apt install qemu-system-arm qemu-user-static fai-server fai-setup-storage fai-doc qemu-utils
# fai-mk-configspace
# export FAI_BASEFILEURL=https://fai-project.org/download/basefiles/
# fai-diskimage -vu armhost -S2G -cDEFAULT,DHCPC,DEBIAN,ARM64,BUSTER_ARM64,FAIBASE,DEMO, CLOUD, LAST arm64.raw
# chown your_user_id arm64.raw
# cp /var/log/fai/armhost/last/vmlinuz* vmlinuz
# cp /var/log/fai/armhost/last/initrd.img* initrd

Then you can run qemu as a normal user:

```
> qemu-system-aarch64 -m 1000 -M virt,gic_version=3 -cpu cortex-a57 -drive file=arm64.raw,
  if=virtio,index=1 -no-reboot -nographic -name ARM64 -net nic,name=eth0,model=virtio -net user,name=eth0,-kernel vmlinuz -initrd initrd -append "console=ttyAMA0 rw ip=dhcp root=/dev/vda1 net.ifnames=0"
```

This works similar for other architectures. Keep in mind, that qemu network setup will have poor performance if not using the virtio driver as above or tap devices.

You can find the base files for many architectures at https://fai-project.org/download/basefiles/, or use mk-basefile to create your own.

## 7.9 FAI rescue system

If you set the variable $FAI_ACTION to sysinfo (for e.g. by using fai-chboot -S), the client will not install a new system, but will collect a lot of system information. If you set $FAI_ACTION to inventory you will only get a few hardware information. Both actions can be used for FAI as a rescue system.

Type `ctrl-c` to get a shell or use Alt-F2 or Alt-F3 and you will get another console terminal, if you have added `createvt` to $FAI_FLAGS.

You now have a running Linux system on the install client without using the local hard disk. Use this as a rescue system if your local disk is damaged or the computer can’t boot properly from hard disk. You will get a shell and you can execute various commands (`dmesg`, `lsmod`, `df`, `lspci`, ...). Look at the log file in `/tmp/fai`. There you can find much information about the boot process.

FAI mounts all file systems it finds on the local disks read only. It also tells you on which partition a file `/etc/fstab` exists. When only one file system table is found, the partitions are mounted according to this information. Here’s an example:

```
dehost:~# df
Filesystem 1K-blocks Used Available Use% Mounted on
devtmpfs   1950880 0 1950880 0% /dev
tmpfs      1971656 336 1971320 1% /run
overlay    1971656 148 1971508 1% /live/cow
192.168.33.250:/srv/fai/nfsroot 15416320 12121088 919552 93% /live/image
faiserver:/srv/fai/config 15416320 12121088 919552 93% /var/lib/fai/config
/dev/mapper/vg1-root 7828376 1599880 5811116 22% /target
/dev/udal 198337 49976 138121 27% /target/boot
/dev/mapper/vg1-home 1503280 4620 1404248 1% /target/home
```

This method can be used as a rescue environment! If you need a file system with read-write access use the `rwmount` command:
7.10 FAI without NFS

To boot into FAI and begin the installation sequence without using the NFS protocol, you boot the client machine using PXE as usual and then retrieve an image containing the nfsroot via http.

To create an image, use fai-cd’s -S argument and -M to avoid setting up a partial mirror within the image.

```
faiserver# fai-cd -M -S squash.img
```

Move this image to a directory from which it can be requested via http (usually a directory served by the webserver).

To now request the squashfs image, add the following to your kernel command line, e.g. in your pxelinux configuration file for the client with fai-chboot.

```
root=live:http://faiserver/cskoeln/squash.img FAI_CONFIG_SRC=file:///var/lib/fai/config
```

Replace faiserver with the domain name or IP of the machine your squash image is served from.

7.11 Installing other distributions using a Debian nfsroot

You can install all sorts of Linux distributions from a single Debian nfsroot. Therefore you have to create a base.tar.xz of the distribution you like to install and place it into the basefiles directory. Then name it UBUNTU1404.tar.xz for example. An install client which belongs to the class UBUNTU1404 then extracts this base file into its empty file system. Additionally you have to adjust the sources.list or similar configuration files which are needed for specifying the location of the package repository.

The tool rinse(8) is used for creating base files for distribution like CentOS, openSUSE, Scientific Linux Cern or Fedora. Some basefiles can be downloaded from https://fai-project.org/download/basefiles/.

The script mk-basefile in /usr/share/doc/fai-doc/examples/simple/basefiles/ helps creating this base files.

7.12 Creating chroot and virtualization environments

If you have to create some chroot environments, or a virtualization environment where you neither can nor want to run a normal Debian Installer in to get to a working system (for example, Xen guest domains), there is the FAI action dirinstall. By calling

```
faiserver# fai <options> dirinstall <target-directory>
```

and using either the option -c <classes> or -N you get a FAI installation, without the partitioning action, right into the target directory. The host name for the target installation can be specified using -u <host-name>

This, for example, can be used to combine FAI with the tool xen-tools, which helps you to build Xen guest domains. xen-tools are very nice for generating configuration files and block devices for new guests based on simple commands and/or configuration files, but they can only assign one role per installation for customization. FAI-users need and want more, as they are used to have the class system. They get them even in xen-tools installations, by using the following code as a xen-tools role script:

```
#!/bin/sh
TARGET=$1
CMD="fai -N -v -u ${hostname} dirinstall $TARGET"
echo running $CMD
$CMD
```

Then, you should set the variable install=0 in the config of xen-tools for that host.
7.13 Using FAI for updates

FAI can also do updates of already running systems, without a re-installation from scratch. This is called softupdate. A FAI softupdate skips the tasks which are not suitable for updating a running system, like partitioning the hard disks and creating file systems. Instead, it only executes the tasks for updating and installing software packages and calling the customization scripts.

To run a softupdate call:

```
# fai -v -s nfs://faiserver/srv/fai/config softupdate
```

By default, a softupdate uses the list of classes defined during the initial installation. Make sure to set the variable $LOGSERVER (done in a class/*.var file) if FAI should save the log files to a remote machine.

It’s up to you, how to start a softupdate on a bigger number of hosts. You may do the softupdate on a regular basis via cron or you can use tools like clusterssh(1) to start a softupdate via a push on a list of hosts.

Keep in mind, that the customization scripts are run every time you do a softupdate. That means, they have to be idempotent i.e. the result of their operation should always produce the same result, even when they run more than once.

For example appending a line to a file must not be done via this code:

```
$ echo "some strings" >> /etc/fstab
```

Instead use the command ainsl(1) in a shell script or use cfengine’s function AppendIfNoSuchLine.

All commands in the customization script must be capable of modifying the target file system whether it’s available in /target during the initial installation or whether it’s the normal file system relative to / during softupdate.

Here are some variables that help writing these scripts:

- `$target` Points to the root directory of the client, which is /target during installation and / during a softupdate.
- `$FAI_ROOT` It’s the same value as $target. For historic reasons we have both these variables in FAI.
- `$ROOTCMD` In case of the installation this is an alias for chroot $target in case of softupdate it’s just empty. You can prepend this to commands if you need to run a command inside the clients target file system via chroot.
- `$FAI_ACTION` If you need to call code depending on the FAI action performed, you can use this variable. It contains the currently executed action: install, softupdate, dirinstall, sysinfo, inventory or your own defined action.

7.14 How to install 32-bit OS from a 64-bit OS

To install a computer with a 32-bit OS, you need an i386 nfsroot. Creating this 32-bit nfsroot on an install server running amd64 is quite simple. Install and set up the FAI packages. Then copy your FAI config files to a new subdirectory.

```
faiserver# cp -a /etc/fai /etc/fai-i386
```

Edit the variable $FAI_DEBOOTSTRAP_OPTS in /etc/fai-i386/nfsroot.conf and add the option --arch i386. Also choose a different directory for your new nfsroot. Here are the two lines after editing.

```
NFSROOT=/srv/fai/nfsroot-i386
FAI_DEBOOTSTRAP_OPTS="--arch i386 --exclude-info --include=aptitude"
```
Now call fai-make-nfsroot which creates the 32-bit nfsroot in /srv/fai/nfsroot-i386

```
faiserver# fai-make-nfsroot -v -C/etc/fai-i386
```

Creating a partial mirror using `fai-mirror(1)` that is needed for a bootable CD or USB stick is also possible on a different architecture. You have to specify the architecture when calling `fai-mirror`.

```
$ fai-mirror -m800 -B -a i386 -v -cDEFAULT,DEBIAN,FAIBASE,I386 /srv/mirror-i386
```

That’s all!

### 7.15 Aborting the installation when an error occurs

Every task and hook can call the function `task_error` to send an error value to the installation. This error will appear in the log file and can be shown in the `fai-monitor-gui(1)`. The error value is also checked against the variable `$STOP_ON_ERROR` which is set to 700 by default. If the error value is greater than `$STOP_ON_ERROR` the installation will stop immediately. In a hook, just add a call like this:

```
task_error <value>
```

It is comfortable to call `task_error` with ` $?` as second parameter, for e.g. if you want to set an error value of 123 use this

```
<some command>
task_error 123 $? 
```

Then the task error is only set if the preceding command failed with some error. The same syntax can be used by the scripts in the class subdirectory ending in .sh.

A special case are scripts inside the class/ subdirectory that does not end in .sh. In those scripts you have to call `task_error` with 1 as third parameter. As an example you can use those two variants for setting the value to 123

```
task_error 123 $? 1
```

or

```
task_error 123 1 1
```

The latter command always sets the task error to 123 regardless of the value of ` $?`. If the error value is greater than `$STOP_ON_ERROR`, the installation will abort immediately.

The error values are grouped into four categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal or info messages</td>
<td>1xx, 2xx</td>
</tr>
<tr>
<td>warnings</td>
<td>3xx, 4xx</td>
</tr>
<tr>
<td>minor errors</td>
<td>5xx, 6xx</td>
</tr>
<tr>
<td>errors</td>
<td>7xx, 8xx</td>
</tr>
</tbody>
</table>

The fai-monitor-gui has 4 different icons for these error categories.
8 Various hints and details

8.1 The list of tasks

Most tasks of the installation are defined as subroutines which are defined in /usr/lib/fai/subroutines (e.g. task_instsoft). Some are external shell scripts located in /usr/lib/fai/. They are called via a superior subroutine called task. This subroutine calls hooks if available and then calls the task (defined as task_<name>). A task and its hooks can be skipped on demand by using the command skiptask().

Now follows the description of all tasks, listed in the order they are executed.

confdir
The kernel appended parameters may define variables, the syslog daemon is started. Network parameters are fetched from a DHCP server and are stored in boot.log. The DNS resolver configuration file is created.

The location of the configuration space is defined by the variable $FAI_CONFIG_SRC.

After that, the file $FAI/hooks/subroutines is sourced if it exists. Using this file, you can define your own subroutines or override the definition of FAI’s subroutines.

setup
This task sets the system time, all $FAI_FLAGS are defined and two additional virtual terminals are opened on demand. A secure shell daemon is started on demand for remote logins.

defclass
Calls fai-class(1) to define classes using scripts and files in $FAI/class and classes from /tmp/fai/additional-classes and the variable $ADDCLASSES. The list of all defined classes is stored in the variable $classes and saved to /tmp/fai/-FAI_CLASSES.

defvar
Sources all files $FAI/class/*.var for every defined class. If a hook has written some variable definitions to the file $LOGDIR/additional.var, this file is also sourced.

action
Depending on the value of $FAI_ACTION this subroutine decides which action FAI should perform. The default available actions are: sysinfo, install, inventory, dirinstall and softupdate. If $FAI_ACTION has another value, a user defined action is called if a file $FAI/hooks/$FAI_ACTION exists. So you can easily define your own actions.

sysinfo
Called when no installation is performed but the action is sysinfo. It shows information about the detected hardware and mounts the local hard disks read only to /target/partitionname or with regard to a fstab file found inside a partition. Log files are stored to the install server.

inventory
A short list of system information is printed.

install
This task controls the installation sequence. You will hear three beeps before the installation starts. The major work is to call other tasks and to save the output to /tmp/fai/fai.log. If you have any problems during installation, look at all files in /tmp/fai/. You can find examples of the log files at https://fai-project.org/logs/.

dirinstall
Install into a directory, not onto a local disk. Use this for creating chroot environments.

softupdate
This task, executed inside a running system via the fai(8) command line interface, performs a softupdate. See chapter [softupdate] for details.

partition
Calls setup-storage(8) to partition the hard disks and to create file systems. The task writes variable definitions for the root and boot partition and device ($ROOT_PARTITION, $BOOT_PARTITION, $BOOT_DEVICE) to /tmp/-fai/disk_var.sh and creates a fstab file for the new system.
mountdisks
Mounts the created partitions according to the created /tmp/fai/fstab file relative to $FAI_ROOT.

extrbase
Extracts a minimal system after that a chroot can be made into it. By default the base tar file /var/tmp/base.tar.xz will be extracted. Also files matching a class name in $FAI/basefiles/ are used for unpacking a different tar file depending on classes defined. This can be used for installing different Linux distributions than the one used for creating the nfsroot. The default file base.tar.xz is a snapshot of a basic Debian system created by debootstrap(8). This task uses the variable FAI_BASEFILEURL for fetching the base file via FTP, HTTPS or HTTP if it’s defined.

debconf
Calls fai-debconf(1) to set the values for the debconf preseeding database.

repository
Prepare access to the package repository by preparing the apt configuration. This can also add repository keys in a class based manner from files like CLASSNAME.gpg in the directory package_config.

updatebase
Updates the base packages of the new system and updates the list of available packages. It also fakes some commands (called diversions) inside the new installed system using dpkg-divert(8), so no daemons will be started during the installation.

instsoft
Installs the desired software packages using class files in $FAI/package_config/.

configure
Calls scripts in $FAI/scripts/ and its subdirectories for every defined class.

tests
Calls test scripts in $FAI/tests/ and its subdirectories for every defined class.

finish
Unmounts all file systems in the new installed system and removes diversions of files using the command fai-divert.

chboot
Changes the PXE configuration for a host on the install server which indicates which PXELINUX configuration to load on the next boot from network card via TFTP. Therefore the fai-chboot(8) command is executed remotely on the install server.

savelog
Saves log files to local disk and to the account $LOGUSER on $LOGSERVER (defaults to the install server).

faiend
Wait for background jobs to finish (e.g. emacs compiling lisp files) and automatically reboots the install clients or waits for manual input before reboot.

8.2 Automated tests
After the customization scripts are executed, FAI will execute some tests if available. Using these test, you can check for errors of the installation. Test scripts are called via fai-do-scripts(1) and should append its messages to $LOGDIR/test.log. A Perl module including some useful subroutines can be found in Faitest.pm. A test can also define a new class for executing another tests during next boot via the variable $ADDCLASSES.

8.3 Autodiscover
In FAI 5.0 we released a feature that allows clients to search for the faiserver in their respective subnetwork. This lifts the necessity of having to collect every client’s MAC address and configuring the DHCP daemon.

This is done by booting from a small FAI autodiscover bootmedium (CD, USB, etc.), which can be created via the command:
The image is roughly 25MB in size and scans the subnet for a FAI server. By default it shows a menu with all profiles available in the configuration space in the same manner as the menu flag does. From this menu, you can select the installation type you wish to perform.

For the clients to find the faiserver, the faiserver must run fai-monitor.

### 8.4 Changing the boot device

Changing the boot sequence is normally done in the BIOS setup. But you can’t change the BIOS from a running Linux system. So, the boot sequence of the BIOS will remain unchanged and your computer should always boot first from its network card and the second boot device should be the local disk. Then you can change the boot device of the client by creating different PXELINUX configurations. This will define if an installation should be performed, or if the client should to boot from local disk. This is done using fai-chboot(8).

### 8.5 How to use NFS v4 with the nfsroot

The nfsroot is mounted via NFS v3 by default. This is done by adding the option nfsvers=3 to the kernel boot options of the kernel booted by the install clients. fai-chboot(8) will do this for you.

If you want to use NFS v4 for the nfsroot, you have to remove this option the kernel boot options. Additionally, overlayfs - the kernel module that is used for making the read-only mounted nfsroot writeable - can only work with NFS v4 if acl(5) support is disabled on the filesystem on the NFS server side (as of 2019). If /srv/fai/nfsroot on your FAI server is located on the root partition mounted to /, you can disable acl support for the whole file system on this partition using

```
# mount -oremount,noacl /
```

Then mounting the nfsroot from a FAI server using NFS v4 will work.

### 8.6 How to create a local Debian mirror

The utility mkdebmirror can be used for creating your own local Debian mirror. This script uses debmirror(1). A partial Debian mirror for i386 and amd64 architecture for Debian 8.0 (aka jessie) without the source packages needs about 56GB of disk space. Accessing the mirror via HTTP will be the default way in most cases. To see more output from the script call mkdebmirror -v. A root account is not necessary to create and maintain the Debian mirror.

To use HTTP access to the local Debian mirror, install a web server and create a symlink to the local directory where your mirror is located:

```
faiserver# apt-get install apache2
faiserver# ln -s /files/scratch/debmirror /var/www/html/debmirror
```

Create a file sources.list(5) in /etc/fai/apt which gives access to your Debian mirror. Also add the IP-address of the HTTP server to the variable $NFSROOT_ETC_HOSTS in nfsroot.conf if the install clients have no DNS resolving.

---

18 You can find the script in /usr/share/doc/fai-doc/examples/utils/
8.7 Small hints

• When using HTTP access to a Debian mirror, the local /var partition on all install clients must be big enough to keep the downloaded Debian packages. Do not try with less than 250 Mbytes unless you know why. You can limit the number of packages installed at a time with the variable $MAXPACKAGES.

• You can remove the red logo on the install client by simply calling reset once. If will also not appear if you create a file using this command on the install server:

```bash
touch /srv/fai/nfsroot/.nocolorlogo
```

• A list of variables used by FAI can be found at https://wiki.fai-project.org/index.php/Variables.

• You can shorten some customization scripts by using one single fcopy command fcopy -r /

• If you rebuild the nfsroot, you will create a new ssh host key inside the nfsroot. Then logging in to an install client may fail, because the host key changes. You can use this:

```bash
$ ssh -o StrictHostKeyChecking=no root@installclient
```

• You can also delete the host entry on your install client in your ~/.ssh/known_hosts file by using the ssh-keygen -R command.

• In the tasks chboot and savelog, a connection using secure shell is opened to the FAI server (see [isavelog]). To ensure that this works non-interactively, a proper entry in NFSROOT/root/.ssh/known_hosts must be created. When using fai-setup, this is done automatically, but it may require manual editing in case the name of your FAI server was not determined correctly. If you stumble over ssh connections that require typing "yes" to accept the host key during installation, please check the contents of your NFSROOT/root/.ssh/known_hosts file

• A list of all local hard disks is stored in $disklist. It’s defined after set_disk_info is called.

• Use fai-divert -a if a postinst script calls a configuration program, e.g. the postinst script for package apache calls apacheconfig, which needs manual input. You can fake the configuration program so the installation can be fully automatic.

• Sometimes the installation seems to stop, but often there’s only a postinstall script of a software package that requires manual input from the console. Change to another virtual terminal and look which process is running with tools like top(1) and pstree(1). You can add debug to FAI_FLAGS to make the installation process show all output from the postinst scripts on the console and get its input also from the console.

• How can I define classes on the kernel command line?

  Read the man page of fai-class(8). If you like to define some additional classes (for e.g. A,B,C) on the kernel command line add this: ADDCLASSES=A,B,C

• How to use a custom kernel inside the nfsroot?

  Build your customized kernel by building a kernel package using make-kpkg(8) and use the option --initrd. Copy this Debian package to a local repository and add it to /etc/fai/sources.list. Add the name of your package to /etc/fai/NFSROOT. Then call

```bash
# fai-make-nfsroot -k
```

• Can I use a 4.X kernel?

  Yes. Using FAI 5.1 and dracut 044+150-1 overlayfs (instead of aufs) is supported, so is kernel 4.x. When using Debian jessie, you may use a kernel from backports or have a look at https://lists.uni-koeln.de/pipermail/linux-fai/2016-March/011283.html
• How to use the nfsroot as system for diskless clients?
  https://wiki.fai-project.org/index.php/Use_nfsroot_for_diskless_clients

• How to serve multiple nfsroot directories on one FAI server?
  If you want to serve multiple nfsroot directories, you need to create specific config directories in /etc for FAI, like /etc/fai-jessie and /etc/fai-stretch. Then you need to set the $NFSROOT variables to different directories and run

```bash
faiserver#fai-make-nfsroot -C /etc/fai-jessie
```

8.8 flag_reboot (FAI_FLAGS)

If flag_reboot is set, by adding "reboot" to $FAI_FLAGS, your client machine will reboot after the task faiend has finished. This is true for network as well as bootmedium installations.

8.9 CentOS reboot

After the installation CentOS usually requires one additional reboot, because of the SELinux security fixes that are applied post-installation.

8.10 Log files

FAI is creating several log files. During installation they are stored in /tmp/fai on the install client itself. At the end of the installation they will be copied to the install server (see [isavelog]). After the install client rebooted into his newly installed system, you can find the FAI logs in /var/log/fai. Log files are also created when doing the softupdate or dirinstall action.

On the faiserver, you can find the (remote) log files under the ~fai directory.

Sample log files from successfully installed computers are available on https://fai-project.org/logs. These a some log files which are created by FAI.

**FAI_CLASSES**
Contains a list of all classes defined.

**dmesg.log**
Output of the dmesg command. Contains useful messages of the kernel ring buffer.

**fai.log**
The main log file. Contains all important information. You should always read this file.

**boot.log**
A list of variables of network parameters, mostly defined by the DHCP daemon.

**format.log**
Output of the partition tool setup-storage(8).

**shell.log**
Output of all shell scripts, that are used for customization.

**variables.log**
A list of all shell variables which are available during an installation.

**error.log**
A summary of possible errors in all log files.
**disk_var.sh**

A list of variables that contain information about devices and partitions to boot from, the root partition and a list of swap devices. These information is used by some customization scripts (e.g. `GRUB_PC/10-setup`).

If the installation process finishes, the hook `savelog-LAST.sh` searches all log files for common errors and writes them to the file `error.log`. So, you should first look into this file for errors. Also the file `status.log` give you the exit code of the last command executed in a script. To be sure, you should look for more details in all log files.

### 8.11 How to use HTTP for PXE boot

`fai-make-nfssroot` now uses the `lpxelinux.0` binary which already supports transfer of the kernel and initrd via http (additional to tftp). You only have to enable HTTP access to the tftp directory:

```
cd /var/www/html
ln -s /srv/tftp/fai
```

Add `-U URL` to the `fai-chboot` call. For example:

```
fai-chboot -U http://faiserver/fai -IFv ........
```

### 9 Troubleshooting

#### 9.1 Boot errors

The following error message indicates that your install client doesn’t get an answer from a DHCP server. Check your cables or start the `dhcppd(8)` daemon with the debug flag enabled.

```
PXE-E51: No DHCP or BOOTP offers received
Network boot aborted
```

If you do not see the following message, the install kernel could not detect your network card, for example because of a missing driver:

```
Starting dhcp for interface eth0
dhcp: PREINIT eth0 up
dhcp: BOND setting eth
```

Check the `initrd` in the `nfssroot` (`lsinird`) if the kernel driver of your network card is included there and check if you like to add the package `firmware-linux-nonfree` in `/etc/fai/NFSROOT` and rebuild the `initrd` by calling `fai-make-nfssroot -k`. You may also add a driver to `/srv/fai/nfssroot/etc/dracut.conf` in the line `add_drivers+=.`.

This is the error message you will see, when your network card is working, but the install server does not export the `nfssroot` directory to the install clients. This is often caused by missing NFS permissions on the server side.
Starting dhcp for interface eth0
dhcp: PREINIT eth0 up
dhcp: BOND setting eth
mount.nfs: access denied by server while mounting 192.168.33.250:/srv/fai/nfsroot
.
.

dracut Warning: Could not boot
.
Dropping to debug shell
dracut:/#

Now, you are inside the emergency shell of the initrd which was created by `dracut(8)`. You will get a shell prompt, and can look at the log files. For more information about debugging the early boot process using dracut see `dracut.cmdline(7)`

Use the following command on the install server to see which directories are exported from the install server (named faiserver):

```
$ showmount -e faiserver
```