grml-live(8)

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
0.27.2	2016-02-15	Unify duplicated base.tgz documentation, add instructinos how to run grml-live directly from git, provide FAQ entry how to serve a local repository via HTTP.	MP
0.1	2010-09-01	Initial document version with included revision information.	MP

Contents

1 Name

grml-live - build framework based on FAI for generating a Grml and Debian based Linux Live system (CD/ISO)

2 Synopsis

grml-live [-a <architecture>] [-c <classe[s]>] [-C <configfile>] [-e <extract_iso_name>] [-g <grml_name>] [-i <iso_name>] [-o <output_directory>] [-r <release_name>] [-s <suite>] [-t <template_directory>] [-v <version_number>] [-U <username>] [-w <date>] [-AbBFnNqQuVz]

3 Description

grml-live provides the build system for creating a Grml and Debian based Linux Live-CD. The build system is based on FAI (Fully Automatic Installation). grml-live uses the "fai dirinstall" feature to generate a chroot system based on the class concept of FAI (see later sections for further details) and provides the framework to be able to generate a full-featured ISO. It does not use all the FAI features by default though and you don't have to know FAI to be able to use it.

The use of FAI gives you the flexibility to choose the packages you would like to include on your very own Linux Live-CD without having to deal with all the details of a build process.

Caution
grml-live does **not** use /etc/fai for configuration but instead (unless overridden using the *'-D'* option). This ensures that it does not clash with default FAI configuration and packages, so you can use grml-live and FAI completely independent at the same time!

Note
Please notice that you should have a fast network connection as all the Debian packages will be downloaded and installed via network. If you want to use a local mirror (strongly recommended if you plan to use grml-live more than once) check-out mkdebmirror (see /usr/share/doc/grml-live/examples/mkdebmirror), debmirror(1), reprepro(1) (see /usr/share/doc/grml-live/examples/reprepro/ for a sample configuration), apt-cacher(1) and approx(8). To avoid downloading the base system again and again check out the base tar.gz feature.

4 Options

-A
Clean up all output directories before running the build process. After finishing, clean up the Chroot target and Build target directories.

-a ARCHITECTURE
Use the specified architecture instead of the currently running one. This allows building a 32bit system on a 64bit host (though you can't build a 64bit system on a 32bit system/kernel of course). Please notice that real crosscompiling (like building a ppc system on x86) isn't possible due to the nature and the need of working in a chroot. Currently supported values: i386 and amd64.

-b
Build the ISO without updating the chroot via FAI. This option is useful for example when working on stable releases: if you have a working base system/chroot and do not want to execute any further updates (via "-u" option) but intend to only build the ISO.

-B

Build the ISO without touching the chroot at all. This option is useful if you modified anything that FAI or grml-live might adjust via Grml's FAI scripts. It's like the *-b* option but even more advanced. Use only if you really know that you do not want to update the chroot.

-c CLASSES

Specify the CLASSES to be used for building the ISO via FAI. By default only the classes GRMLBASE, GRML_FULL and I386/AMD64 (depending on system architecture) are assumed, resulting in a base system of about 350MB total ISO size. If using a non-I386 system (like AMD64) you should specify the appropriate architecture as well. Additionally you can specify a class providing a grml-kernel (see the CLASSES section in this document for details about available classes). So instead of GRML_FULL you can also use GRML_SMALL and GRML_FULL.

Important

All class names should be written in uppercase letters. Do not use a dash, use an underscore. So do not use "amd64" but "AMD64", do not use "FOO BAR" but "FOO_BAR".

-C CONFIGURATION_FILE

The specified file is used as configuration file for grml-live. By default /etc/grml/grml-live.conf is used as default configuration. If a file named /etc/grml/grml-live.local exists it is used as well (sourced after reading /etc/grml/grml-live.conf meant as main file for local configuration). As a last option the specified configuration file is sourced so it is possible to override settings of /etc/grml/grml-live.conf as well as of /etc/grml/grml-live.local. Please notice that all configuration files have to be adjusted during execution of grml-live, so please make sure you use /etc/grml/grml-live.conf as a base for your own configuration file (usually /etc/grml/grml-live.local). Please also notice that the configuration file specified via this option is **not** (yet) /etc/grml/grml-live.local for configuration stuff used inside

-d DATE

Use specified date as build date information on the ISO instead of the default. The default is the date when grml-live is being executed (retrieved via executing *date +%Y-%m-%d*). The information is stored inside the file /GRML/grml-version on the ISO, /etc/grml_version in the squashfs file and in all the bootsplash related files. This option is useful if you want to provide an ISO with release information for a specific date but have to build it in advance. Usage example: *-d 2009-10-30*

-D CONFIGURATION_DIRECTORY

The specified directory is used as configuration directory for grml-live and its FAI. By default /etc/grml/fai is used as default configuration directory. If you want to have different configuration scripts, package definitions, etc. with without messing with the global configuration under /etc/grml/fai provided by grml-live this option provides you the option to use your own configuration throughout this documentation.

-e EXTRACT_ISO_NAME

The squashfs inside the specified ISO will be extracted and used as the chroot. This option is useful for remastering, in combination with -A and -b or -u.

-F

Force execution and do not prompt for acknowledgment of configuration.

-g GRML_NAME

Set the grml flavour name. Common usage examples: grml, grml-small, grml64. Please do NOT use blanks and any special characters like */*, *;* inside GRML_NAME, otherwise you might notice problems while booting.

-h

Display short usage information and exit.

-i ISO_NAME

Specify name of ISO which will be available inside $OUTPUT_DIRECTORY/grml_isos by default.

-I CHROOT_INSTALL

Specify name of source directory which provides files that should become part of the chroot/ISO. Not enabled by default. Note: the files are installed under */* in the chroot so you have to create the rootfs structure on your own.

-n
Skip creation of the ISO file. This option is useful if you want to build/update the chroot and/or recreate the squashfs file without building an ISO file.

-N
Bootstrap the chroot without building bootloader, squashfs, or finalizing the ISO. Use this option if installation of some packages fails, you want to run custom commands or similar. The main use of this option is to save time by skipping stages which aren't necessary for bootstrapping the chroot and which would get executed more than once when iterating through the initial bootstrapping. Alternatively, use this option as a test run of grml-live. Once you are satisfied with the state of your grml_chroot, use grml-live **-u** to build the remaining stages and finalize the ISO.

-o OUTPUT_DIRECTORY
Main output directory of the build process of FAI. Some directories are created inside this target directory, being: grml_cd (where the files for creating the ISO are located, including the compressed squashfs file), grml_chroot (the chroot system) and grml_isos (where the resulting ISO is stored).

-q
Build the ISO without (re-)creating the squashfs compressed file using mksquashfs. This option is useful if you just want to update parts outside the chroot in the ISO. Consider combining this option with the build-only option *-b*.

-Q
Build the ISO without generating a netboot package.

-r RELEASENAME
Specify name of the release.

-s SUITE
Specify the Debian suite you want to use for your live-system. If unset defaults to "testing". Supported values are: stable, testing, unstable (or their corresponding release names like "jessie"). Please be aware that recent Debian suites might require a recent base.tgz debootstrap.

-t TEMPLATE_DIRECTORY
Specify place of the templates used for building the ISO. By default (and if not manually specified) this is /usr/share/grml-live/templates/.

-u
Update existing chroot instead of rebuilding it from scratch. This option is based on the softupdate feature of FAI.

-U USERNAME
Sets ownership of all build output files to specified username before exiting.

-v VERSION_NUMBER
Specify version number of the release.

-V
Increase verbosity in the build process.

-w DATE
The wayback machine. Build the system using Debian archives from the specified date. Valid date formats are yyyymmdd-dThhmmssZ or simply yyyymmdd. To learn which snapshots exist, i.e. which date strings are valid, simply browse the lists on http://snapshot.debian.org/. If there is no import at the exact time you specified you will get the latest available timestamp which is before the time you specified. This option is useful especially for release and debugging builds - for example if you know that the Debian archive was in a good state on a specific date but you want to build it on another day in the future, where the archive might not be as needed anymore. Please be aware that this is restricted to the plain Debian repositories only, as referred to in /etc/apt/sources.list.d/debian.list (so neither the Grml repositories nor any further custom ones are affected by the wayback machine).

-z
Use ZLIB instead of LZMA/XZ compression in mksquashfs part of the build process.

5 Usage examples

To get a Debian-stable and Grml-based Live-CD using /grml/grml-live as build and output directory just run:

```
# grml-live
```

To get a 64bit Debian-testing and grml-small based Live-CD using /srv/grml-live as build and output directory use the following command line on your amd64 system:

```
# grml-live -s testing -c GRMLBASE,GRML_SMALL,AMD64 -o /srv/grml-live
```

Note
If you have enough RAM, just run "mount -t tmpfs none /media/ramdisk" to get a tmpfs ("RAMDISK"), and use /media/ramdisk as build and output directory - this results in a very fast build process. Note that these files will be gone when rebooting.

---

6 Main features of grml-live

- create a Grml-/Debian-based Linux Live-CD with one single command
- class based concept, providing a maximum of flexibility
- supports integration of own hooks, scripts and configuration
- supports use and integration of own Software and/or Kernels via simple use of Debian repositories
- native support of FAI features

7 The class concept

grml-live uses FAI and its class based concept for adjusting configuration and setup according to your needs. This gives you flexibility and strength without losing the simplicity in the build process.

The main and base class provided by grml-live is named GRMLBASE. It's strongly recommended to **always** use the class GRMLBASE when building an ISO using grml-live, as well as the architecture dependent class which provides the kernel (being *I386* for x86_32 and *AMD64* for x86_64) and a GRML_* class (like GRML_SMALL or GRML_FULL). The following files and directories are relevant for class GRMLBASE by default:

```
${GRML_FAI_CONFIG}/config/scripts/GRMLBASE/
${GRML_FAI_CONFIG}/config/debconf/GRMLBASE
${GRML_FAI_CONFIG}/config/class/GRMLBASE.var
${GRML_FAI_CONFIG}/config/hooks/instsoft.GRMLBASE
${GRML_FAI_CONFIG}/config/package_config/GRMLBASE
```

Take a look at the next section for information about the concept of those files/directories.

If you want to use your own configuration, extend an existing configuration and/or add additional packages to your ISO just invent a new class (or extend an existing one). For example if you want to use your own class named "FOOBAR" just set CLASSES="GRMLBASE,GRML_SMALL,AMD64,FOOBAR" inside /etc/grml/grml-live.local or invoke grml-live using the classes option: "grml-live -c GRMLBASE,GRML_SMALL,AMD64,FOOBAR ...".

More details regarding the class concept can be found in the documentation of FAI itself (being available at /usr/share/doc/fai-doc/).

8 Available classes

The package selection part of the classes can be found in selected. The following classes are predefined:

- DEBORPHAN: get rid of all packages listed in output of deborphan
- FRESHCLAM: execute freshclam (if it's present) to update clamav definitions (increases resulting ISO size ~70MB). By default it's skipped to avoid bigger ISO size.
- GRMLBASE: the main class responsible for getting a minimal subset of what's defining a Grml system. Important parts of the buildprocess are specified in this class as well, so unless you have a really good reason you should always use this class.
- GRML_FULL: full featured Grml, also known as the "normal", full grml as introduced in December 2011 (~460MB ISO size).
- GRML_SMALL: minimum sized Grml version, known as grml-small (~230MB ISO size).
- LATEX: LaTeX(-related) packages like auctex, texlive,... (which used to be shipped by grml before the LaTeX removal)
- LATEX_CLEANUP: get rid of several very large LaTeX directories (like some /usr/share/doc/texlive-*, /usr/share/doc/texmf,...)
- LOCALES: use full featured locales setup (see /etc/locale.gen.grml). This avoids to get rid of /usr/share/locale - which happens by default otherwise - as well.
- NO_ONLINE: do not run scripts during the chroot build process which require a network connection
- RELEASE: run some specific scripts and commands to provide the workflow for an official grml release
- REMOVE_DOCS: get rid of documentation directories (like /usr/share/doc, /usr/share/man/, /usr/share/info,...)
- SOURCES: retrieve Debian source packages after installation. Files will be placed in the output directory under grml_sources.
- XORG: providing important packages for use with a base grml-featured X.org setup

9 Files

Notice that grml-live ships FAI configuration files that do not use the same namespace as the FAI packages itself. This ensures that grml-live does not clash with your usual FAI configuration, so instead of /etc/fai/fai.conf (package below. To get an idea how another configuration or example files could look like check out /usr/share/doc/fai-doc/examples/simple/ (provided by Debian package fai-doc). Furthermore /usr/share/doc/fai-doc/fai-guide.html/ch-config.html provides documentation regarding configuration possibilities.

`/usr/sbin/grml-live`

Script for the main build process. Requires root permissions for execution.

`/etc/grml/grml-live.conf`

Main configuration file for grml-live which should be considered as a reference configuration file only. Please use /etc/grml/grml-live.local for local configuration instead.

`/etc/grml/grml-live.local`

All the local configuration should go to this file. This file overrides any defaults of grml-live. Configurations via /etc/grml/grml-live.local are preferred over the ones from /etc/grml/grml-live.conf. If you want to override settings from /etc/grml/grml-live.local as well you have to specify them on the grml-live commandline.

`${GRML_FAI_CONFIG}/fai.conf`

Main configuration file for FAI which specifies where all the configuration files and scripts for FAI/grml-live can be found. By default the configuration variables are FAI_CONFIG_SRC=file:///etc/grml/fai/config and GRML_FAI_CONFIG=/etc/grml/fai/config - both pointing to a directory shipped by grml-live out-of-the-box so you shouldn't have to configure anything in this file.

`${GRML_FAI_CONFIG}/config/`

The main directory for configuration of FAI/grml-live. More details below.

`${GRML_FAI_CONFIG}/config/class/`

This directory contains files which specify main configuration variables for the FAI classes.

`${GRML_FAI_CONFIG}/config/debconf/`

This directory provides the files for preseeding/configuration of debconf through files.

`${GRML_FAI_CONFIG}/config/hooks/`

This directory provides files for customising the build process through hooks. Hooks are user defined programs or scripts, which are called during the installation process.

`${GRML_FAI_CONFIG}/config/package_config/`

Directory with lists of software packages to be installed or removed. The different classes describe what should find its way to your ISO. When running "grml-live -c GRMLBASE,GRML_SMALL,AMD64 ..." only the configuration of GRMLBASE, GRML_SMALL and and AMD64 will be taken. If you use *grml-live -c GRMLBASE,GRML_SMALL,AMD64,FOOBAR ...* then the files of GRMLBASE, GRML_SMALL, AMD64 **plus** the files from FOOBAR will be taken. So just create a new class to adjust the package selection according to your needs. Please notice that the directory GRMLBASE contains a package list defining a minimum but still reasonable package configuration.

`${GRML_FAI_CONFIG}/config/scripts/`

Scripts for customising the ISO within the build process.

`${GRML_FAI_CONFIG}/live-initramfs/`

This directory provides the files used for building the initramfs/initrd via live-initramfs(8).

10 Available log files

Starting with grml-live version 0.17.0 you should find log files in a directory named *grml_logs* in the output directory (next to grml_isos, grml_chroot,...).

grml-live versions before 0.17.0 used to log into /var/log/grml-live.log and /var/log/fai/grml.

11 Requirements for the build system

- any Debian based system should be sufficient (if it doesn't work it's a bug, please send us a bug report then). Check out How do I deploy grml-live on a plain Debian installation for details how to set up grml-live on a plain, original Debian system.
- enough free disk space; at least 800MB are required for a minimal grml-live run (~400MB for the chroot [$CHROOT_OUTPUT], ~150MB for the build target [$BUILD_OUTPUT] and \~150MB for the resulting ISO [$ISO_OUTPUT] plus some temporary files), if you plan to use GRML_FULL you should have at least 4GB of total free disk space
- fast network access for retrieving the Debian packages used for creating the chroot (check out "local mirror" to workaround this problem as far as possible)

For further information see next section.

12 Current state of grml-live with squashfs-tools and kernel

Use squashfs-tools >=4.2-1 to build Grml (based) ISOs featuring kernel version 2.6.38-grml[64] or newer.

13 FAQ

13.1 How do I deploy grml-live on a plain Debian installation?

The easiest way to get a running grml-live setup is to just use Grml. Of course using grml-live on a plain, original Debian installation is supported as well. So there we go.

What we have: plain, original Debian jessie (8.x).

What we want: build a Grml ISO based on Debian/jessie for the amd64 architecture using grml-live.

13.1.1 Instructions

```
# adjust sources.list:
cat >> /etc/apt/sources.list << EOF

# grml stable repository:
  deb     http://deb.grml.org/ grml-stable  main
# deb-src http://deb.grml.org/ grml-stable  main

# grml testing/development repository:
  deb     http://deb.grml.org/ grml-testing main
# deb-src http://deb.grml.org/ grml-testing main
EOF

# get keyring for apt:
apt-get update
apt-get --allow-unauthenticated install grml-debian-keyring

# optionally(!) install basefile so we don't have to build basic
# chroot from scratch, grab from http://daily.grml.org/
# mkdir -p /etc/grml/fai/config/basefiles/
# mv I386.tar.gz /etc/grml/fai/config/basefiles/
# mv AMD64.tar.gz /etc/grml/fai/config/basefiles/

# install relevant tools
apt-get --no-install-recommends install grml-live

# adjust grml-live configuration for our needs:
cat > /etc/grml/grml-live.local << EOF
## want a faster build process and don't need smaller ISOs?
## if so use zlib compression
# SQUASHFS_OPTIONS="-comp gzip -b 256k"
## want to use a specific squashfs binary?
# SQUASHFS_BINARY='/usr/bin/mksquashfs'
## install local files into the chroot
# CHROOT_INSTALL="/etc/grml/fai/chroot_install"
## adjust if necessary (defaults to /grml/grml-live):
## OUTPUT="/srv/grml-live"
```

```
FAI_DEBOOTSTRAP="jessie http://ftp.debian.org/debian/"
# ARCH="amd64"
CLASSES="GRMLBASE,GRML_FULL,AMD64"
EOF

# just optional(!) - upgrade FAI to latest available version:
cat >> /etc/apt/sources.list.d/fai.list << EOF
deb     http://jenkins.grml.org/debian fai main
deb-src http://jenkins.grml.org/debian fai main
EOF

# get gpg key of FAI repos and install current FAI version:
wget -O - http://jenkins.grml.org/debian/C525F56752D4A654.asc | sudo apt-key add -
apt-get update
apt-get install fai-client fai-server fai-doc
```

That's it. Now invoking *grml-live -V* should build the ISO. If everything worked as expected the last line of the shell output should look like:

```
[*] Successfully finished execution of grml-live [running 687 seconds]
```

and the ISO can be found inside /grml-live/grml-live/grml_isos/ then.

13.2 What is $GRML_FAI_CONFIG?

The variable *$GRML_FAI_CONFIG* is pointing to the directory /etc/grml/fai by default. To provide you a maximum of flexibility you can set up your own configuration directory (e.g. based on /etc/grml/fai) and use this directory running grml-live with the *-D <config_dir>* option. Now *$GRML_FAI_CONFIG* points to the specified directory instead of using /etc/grml/fai and all the configuration files, scripts and hooks will be taken from your *$GRML_FAI_CONFIG* directory.

13.3 I've problems with the build process. How to start debugging?

Check out the logs inside the directory *grml_logs* next to your grml_chroot, grml_isos,... directories.

If you need help with grml-live or would like to see new features as part of grml-live you can get commercial support via Grml Solutions.

13.4 How do I install further files into the chroot/ISO?

Just point the configuration variable CHROOT_INSTALL to the directory which provides the files you would like to install. Note that the files are installed under / in the chroot - so you have to create the rootfs structure on your own. Usage example:

```
echo "CHROOT_INSTALL=\$GRML_FAI_CONFIG/chroot_install" >> /etc/grml/grml-live. ↩
   local
mkdir -p /etc/grml/fai/chroot_install/usr/src/
wget example.org/foo.tar.gz
mv foo.tar.gz /etc/grml/fai/chroot_install/usr/src/
grml-live ...
```

13.5 Can I use my own (local) Debian mirror?

Yes. Set up an according sources.list configuration as class file in FAI_DEBOOTSTRAP (if not already using a base.tgz) inside /etc/grml/grml-live.conf[.local]. If you're setting up your own class file don't forget to include the class name in the class list (grml-live -c ...).

If you want to use a local (for example NFS mount) mirror additionally then adjust MIRROR_DIRECTORY in /etc/grml/grml-live.conf[.local] as well.

If you want to use a HTTP Proxy (like apt-cacher-ng), set APT_PROXY. Example:

```
APT_PROXY="http://localhost:3142/"
```

13.6 How do I add additional Debian package(s) to my CD/ISO?

Just create a new class (using the package_config directory):

```
# cat > /etc/grml/fai/config/package_config/MIKA << EOF
PACKAGES aptitude


vim
another_name_of_a_debian_package
and_another_one
EOF
```

and specify it when invoking grml-live then:

```
# grml-live -c GRMLBASE,GRML_SMALL,AMD64,MIKA
```

13.7 I fscked up my grml-live configuration. How do I reset it to the defaults?

Notice: this deletes all your grml-live configuration files. If that's really what you are searching for just run:

```
rm -rf /etc/grml/fai /etc/grml/grml-live.conf
dpkg -i --force-confnew --force-confmiss /path/to/grml-live_..._all.deb
```

13.8 Set up apt-cacher-ng for use with grml-live

Make sure /etc/grml/grml-live.local provides according APT_PROXY and FAI_DEBOOTSTRAP:

```
# cat /etc/grml/grml-live.local
[...]
APT_PROXY="http://localhost:3142/"
[...]
FAI_DEBOOTSTRAP="jessie http://localhost:3142/ftp.debian.org/debian jessie main ↩
    contrib non-free"
```

Make sure apt-cacher-ng is running (*/etc/init.d/apt-cacher-ng restart*). That's it. All downloaded files will be cached in /var/cache/apt-cacher-ng then.

13.9 How do I revert the manifold feature from an ISO?

The so called manifold feature Grml ISOs use by default allows one to use the same ISO for CD boot and USB boot. If you notice any problems when booting just revert the manifold feature running:

```
% dd if=/dev/zero of=grml.iso bs=512 count=1 conv=notrunc
```

To switch from manifold to isohybrid mode (an alternative approach provided by syslinux) then just execute:

```
% isohybrid grml.iso
```

13.10 How do I create a base tar.gz (I386.tar.gz or AMD64.tar.gz)

First of all create the chroot using debootstrap (requires root):

```
BASECHROOT='/tmp/basefile'  # path where the chroot gets generated
SUITE='jessie'              # using the current stable release should always work
debootstrap --exclude=info,tasksel,tasksel-data "$SUITE" "$BASECHROOT" http://ftp. ↩
   debian.org/debian
tar -C "$BASECHROOT" --exclude='var/cache/apt/archives/*.deb' -zcf "${SUITE}".tar. ↩
   gz ./
```

Tip
By default debootstrap builds a chroot matching the architecture of the running host system. If you're using an amd64 system and want to build an i386 base.tgz then invoke debootstrap using the *--arch i386* option. Disclaimer: building an AMD64 base.tgz won't work if you are using a 32bit kernel system of course.

Finally place the generated tarball in /etc/grml/fai/config/basefiles/ (note that it needs to be uppercase letters matching the class names, so: e.g. AMD64.tar.gz for amd64 and I386.tar.gz for i386).

Then executing grml-live should use this file as base system instead of executing debootstrap. Check out the output for something like:

```
[...]
ftar: extracting //etc/grml/fai/config/basefiles///AMD64.tar.gz to /srv/ ↩
   grml64_testing/grml_chroot//
[...]
```

13.11 How to use your own local repository

Let's assume you have Debian package(s) in your filesystem inside `/home/foobar/local-packages` and want to provide them to your grml-live build. This can be achieved either 1) through a bind mount (using the MIRROR_DIRECTORY configuration) or 2) by serving a repository via HTTP.

13.11.1 Serving via bind mount / MIRROR_DIRECTORY

Make sure to create an according sources.list configuration file, e.g. using your own class name `CUSTOM`:

```
# cat > /etc/grml/fai/config/files/etc/apt/sources.list.d/local-packages.list/ ↩
   CUSTOM << EOF
deb file:///home/foobar/local-packages ./
EOF
```

Add the according MIRROR_DIRECTORY configuration to your grml-live configuration:

```
# echo "MIRROR_DIRECTORY='/home/foobar/packages'" >> /etc/grml/grml-live.local
```

Make sure the local directory looks like a mirror:

```
% cd /home/foobar/packages
% dpkg-scanpackages . /dev/null | gzip > Packages.gz
```

Finally invoke grml-live with your class name (`CUSTOM` in this example) added to the list of classes on the command line (see grml-live option `-c`).

13.11.2 Serving a repository via HTTP

Make sure to create an according sources.list configuration file, e.g. using your own class name `CUSTOM`:

```
# cat > /etc/grml/fai/config/files/etc/apt/sources.list.d/local-packages.list/ ↩
    CUSTOM << EOF
deb http://127.0.0.1:8000/ ./
EOF
```

Make sure the local directory is served via HTTP on the according IP address and port. For the `http://127.0.0.1:8000/` example from above it should be enough to just invoke:

```
% cd /home/foobar/packages
% dpkg-scanpackages . /dev/null | gzip > Packages.gz
% python -m SimpleHTTPServer 8000
```

Tip

Of course you can also use a real Debian repository setup using tools like reprepro(1) and/or using a real web server, though for quick debugging sessions python's SimpleHTTPServer in combination with dpkg-scanpackages from package dpkg-dev is a simple and easy approach.

Finally invoke grml-live with your class name (`CUSTOM` in this example) added to the list of classes on the command line (see grml-live option `-c`).

14 Download / install grml-live as a Debian package

Stable Debian packages are available through the grml-repository at deb.grml.org and the latest Git commits are available as Debian packages from jenkins.grml.org. If you want to build a Debian package on your own (using for example a specific version or the current development tree), just execute:

```
git clone git://git.grml.org/grml-live
cd grml-live
debuild -us -uc
```

15 Run grml-live directly from git

In case you want to run grml-live directly from the git repository checkout (after making sure all dependencies are installed), you should set `GRML_FAI_CONFIG` so that a) it finds the according FAI configuration files and b) does not use the config files of an possibly installed `grml-live` package. Usage example:

```
# export GRML_FAI_CONFIG=$(pwd)/etc/grml/fai
# export SCRIPTS_DIRECTORY=$(pwd)/scripts
# ./grml-live -s sid -a amd64 -c GRMLBASE,GRML_FULL,AMD64
```

16 Source

The source of grml-live is available at https://github.com/grml/grml-live/

17 Bugs

Please report feedback, bugreports and wishes to the Grml team!

18 Documentation

The most recent grml-live documentation is available online at http://grml.org/grml-live/ and for offline reading also available in different formats:

- http://grml.org/grml-live/grml-live.epub
- http://grml.org/grml-live/grml-live.pdf

19 Authors

Michael Prokop <mika@grml.org>