


Chapter 4. Upgrades from Debian 7 (wheezy)

Table of Contents

4.1. Preparing for the upgrade

We suggest that before upgrading you also read the information in Chapter 5, *Issues to be aware of for jessie*. That chapter covers potential issues which are not directly related to the upgrade process but could still be important to know about before you begin.

4.1.1. Back up any data or configuration information

Before upgrading your system, it is strongly recommended that you make a full backup, or at least back up any data or configuration information you can't afford to lose. The upgrade tools and process are quite reliable, but a hardware failure in the middle of an upgrade could result in a severely damaged system.

The main things you'll want to back up are the contents of **/etc**, **/var/lib/dpkg**, **/var/lib/apt/extended_states** and the output of **dpkg --get-selections "*"** (the quotes are important). If you use **aptitude** to manage packages on your system, you will also want to back up **/var/lib/aptitude /pkgstates**.

The upgrade process itself does not modify anything in the **/home** directory. However, some applications (e.g. parts of the Mozilla suite, and the GNOME and KDE desktop environments) are known to overwrite existing user settings with new defaults when a new version of the application is first started by a user. As a precaution, you may want to make a backup of the hidden files and directories (“dotfiles”) in users' home directories. This backup may help to restore or recreate the

old settings. You may also want to inform users about this.

Any package installation operation must be run with superuser privileges, so either log in as **root** or use **su** or **sudo** to gain the necessary access rights.

The upgrade has a few preconditions; you should check them before actually executing the upgrade.

4.1.2. Inform users in advance

It's wise to inform all users in advance of any upgrades you're planning, although users accessing your system via an **ssh** connection should notice little during the upgrade, and should be able to continue working.

If you wish to take extra precautions, back up or unmount the **/home** partition before upgrading.

You will have to do a kernel upgrade when upgrading to jessie, so a reboot will be necessary. Typically, this will be done after the upgrade is finished.

4.1.3. Prepare for downtime on services

There might be services that are offered by the system which are associated with packages that will be included in the upgrade. If this is the case, please note that, during the upgrade, these services will be stopped while their associated packages are being replaced and configured. During this time, these services will not be available.

The precise downtime for these services will vary depending on the number of packages being upgraded in the system, and it also includes the time the system administrator spends answering any configuration questions from package upgrades. Notice that if the upgrade process is left unattended and the system requests input during the upgrade there is a high possibility of services being unavailable[1] for a significant period of time.

If the system being upgraded provides critical services for your users or the network[2], you can reduce the downtime if you do a minimal system upgrade, as described in Section 4.4.4, “Minimal system upgrade”, followed by a kernel upgrade and reboot, and then upgrade the packages

associated with your critical services. Upgrade these packages prior to doing the full upgrade described in Section 4.4.5, “Upgrading the system”. This way you can ensure that these critical services are running and available through the full upgrade process, and their downtime is reduced.

4.1.4. Prepare for recovery

Although Debian tries to ensure that your system stays bootable at all times, there is always a chance that you may experience problems rebooting your system after the upgrade. Known potential issues are documented in this and the next chapters of these Release Notes.

For this reason it makes sense to ensure that you will be able to recover if your system should fail to reboot or, for remotely managed systems, fail to bring up networking.

If you are upgrading remotely via an **ssh** link it is recommended that you take the necessary precautions to be able to access the server through a remote serial terminal. There is a chance that, after upgrading the kernel and rebooting, you will have to fix the system configuration through a local console. Also, if the system is rebooted accidentally in the middle of an upgrade there is a chance you will need to recover using a local console.

Generally we recommend using the *rescue mode* of the jessie Debian Installer. The advantage of using the installer is that you can choose between its many methods to find one that best suits your situation. For more information, please consult the section “Recovering a Broken System” in chapter 8 of the Installation Guide and the Debian Installer FAQ.

If that fails, you will need an alternative way to boot your system so you can access and repair it. One option is to use a special rescue image or a Linux live CD. After booting from that, you should be able to mount your root file system and **chroot** into it to investigate and fix the problem.

4.1.4.1. Debug shell during boot using initrd

The **initramfs-tools** package includes a debug shell[3] in the initrds it generates. If for example the initrd is unable to mount your root file system, you will be dropped into this debug shell which has basic commands available to help trace

the problem and possibly fix it.

Basic things to check are: presence of correct device files in **/dev**; what modules are loaded (**cat /proc/modules**); output of **dmesg** for errors loading drivers. The output of **dmesg** will also show what device files have been assigned to which disks; you should check that against the output of **echo $ROOT** to make sure that the root file system is on the expected device.

If you do manage to fix the problem, typing **exit** will quit the debug shell and continue the boot process at the point it failed. Of course you will also need to fix the underlying problem and regenerate the initrd so the next boot won't fail again.

4.1.4.2. Debug shell during boot using systemd

If the boot fails under systemd, it is possible to obtain a debug root shell by changing the kernel command line. If the basic boot succeeds, but some services fail to start, it may be useful to add **systemd.unit=rescue.target** to the kernel parameters.

Otherwise, the kernel parameter **systemd.unit=emergency.target** will provide you with a root shell at the earliest possible point. However, this is done before mounting the root file system with read-write permissions. You will have to do that manually with:

```
mount -o remount,rw /
```

More information on debugging a broken boot under systemd can be found in the Diagnosing Boot Problems article.

If everything else fails, you might be able to boot via the old sysvinit system. This requires that **sysvinit** is still installed and the binary **/lib/sysvinit/init** is included in your initramfs. If these requirements are met, add **init=/lib/sysvinit/init** on the kernel command-line and it will boot with the sysvinit binary.

4.1.5. Prepare a safe environment for the upgrade

The distribution upgrade should be done either locally from a

textmode virtual console (or a directly connected serial terminal), or remotely via an **ssh** link.

Important

If you are using some VPN services (such as **tinc**) they might not be available throughout the upgrade process. Please see Section 4.1.3, “Prepare for downtime on services”.

In order to gain extra safety margin when upgrading remotely, we suggest that you run upgrade processes in the virtual console provided by the **screen** program, which enables safe reconnection and ensures the upgrade process is not interrupted even if the remote connection process fails.

Important

You should *not* upgrade using **telnet**, **rlogin**, **rsh**, or from an X session managed by **xdm**, **gdm** or **kdm** etc. on the machine you are upgrading. That is because each of those services may well be terminated during the upgrade, which can result in an *inaccessible* system that is only half-upgraded. Use of the GNOME application **update-manager** is *strongly discouraged* for upgrades to new releases, as this tool relies on the desktop session remaining active.

4.2. Checking system status

The upgrade process described in this chapter has been designed for upgrades from “pure” wheezy systems without third-party packages. For the greatest reliability of the upgrade process, you may wish to remove third-party packages from your system before you begin upgrading.

Direct upgrades from Debian releases older than 7 (wheezy) are not supported. Please follow the instructions in the Release Notes for Debian 7 to upgrade to 7 first.

This procedure also assumes your system has been updated to the latest point release of wheezy. If you have not done this

or are unsure, follow the instructions in Section A.1, “Upgrading your wheezy system”.

4.2.1. Review actions pending in package manager

In some cases, the use of **apt-get** for installing packages instead of **aptitude** might make **aptitude** consider a package as “unused” and schedule it for removal. In general, you should make sure the system is fully up-to-date and “clean” before proceeding with the upgrade.

Because of this you should review if there are any pending actions in the package manager **aptitude**. If a package is scheduled for removal or update in the package manager, it might negatively impact the upgrade procedure. Note that correcting this is only possible if your **sources.list** still points to *wheezy* and not to *stable* or *jessie*; see Section A.2, “Checking your sources list”.

To perform this review, launch **aptitude** in “visual mode” and press **g** (“Go”). If it shows any actions, you should review them and either fix them or implement the suggested actions. If no actions are suggested you will be presented with a message saying “No packages are scheduled to be installed, removed, or upgraded”.

4.2.2. Disabling APT pinning

If you have configured APT to install certain packages from a distribution other than stable (e.g. from testing), you may have to change your APT pinning configuration (stored in **/etc/apt/preferences** and **/etc/apt/preferences.d/**) to allow the upgrade of packages to the versions in the new stable release. Further information on APT pinning can be found in apt_preferences(5).

4.2.3. Checking packages status

Regardless of the method used for upgrading, it is recommended that you check the status of all packages first, and verify that all packages are in an upgradable state. The following command will show any packages which have a status of Half-Installed or Failed-Config, and those with any error status.

```
# dpkg --audit
```

You could also inspect the state of all packages on your system using **aptitude** or with commands such as

```
# dpkg -l | pager
```

or

```
# dpkg --get-selections "*" > ~/curr-pkgs.txt
```

It is desirable to remove any holds before upgrading. If any package that is essential for the upgrade is on hold, the upgrade will fail.

Note that **aptitude** uses a different method for registering packages that are on hold than **apt-get** and **dselect**. You can identify packages on hold for **aptitude** with

```
# aptitude search "~ahold"
```

If you want to check which packages you had on hold for **apt-get**, you should use

```
# dpkg --get-selections | grep 'hold$'
```

If you changed and recompiled a package locally, and didn't rename it or put an epoch in the version, you must put it on hold to prevent it from being upgraded.

The “hold” package state for **apt-get** can be changed using:

```
# echo package_name hold | dpkg --set-selections
```

Replace **hold** with **install** to unset the “hold” state.

If there is anything you need to fix, it is best to make sure your **sources.list** still refers to wheezy as explained in Section A.2, “Checking your sources list”.

4.2.4. The proposed-updates section

If you have listed the **proposed-updates** section in your **/etc/apt/sources.list** file, you should remove it from that

file before attempting to upgrade your system. This is a precaution to reduce the likelihood of conflicts.

4.2.5. Unofficial sources

If you have any non-Debian packages on your system, you should be aware that these may be removed during the upgrade because of conflicting dependencies. If these packages were installed by adding an extra package archive in your **/etc/apt/sources.list**, you should check if that archive also offers packages compiled for jessie and change the source line accordingly at the same time as your source lines for Debian packages.

Some users may have *unofficial* backported "newer" versions of packages that *are* in Debian installed on their wheezy system. Such packages are most likely to cause problems during an upgrade as they may result in file conflicts[4]. Section 4.5, "Possible issues during upgrade" has some information on how to deal with file conflicts if they should occur.

4.3. Preparing sources for APT

Before starting the upgrade you must set up **apt**'s configuration file for package lists, **/etc/apt/sources.list**.

apt will consider all packages that can be found via any "**deb**" line, and install the package with the highest version number, giving priority to the first line in the file (thus where you have multiple mirror locations, you'd typically first name a local hard disk, then CD-ROMs, and then HTTP/FTP mirrors).

A release can often be referred to both by its codename (e.g. **wheezy**, **jessie**) and by its status name (i.e. **oldstable**, **stable**, **testing**, **unstable**). Referring to a release by its codename has the advantage that you will never be surprised by a new release and for this reason is the approach taken here. It does of course mean that you will have to watch out for release announcements yourself. If you use the status name instead, you will just see loads of updates for packages available as soon as a release has happened.

4.3.1. Adding APT Internet sources

The default configuration is set up for installation from the

main Debian Internet servers, but you may wish to modify **/etc/apt/sources.list** to use other mirrors, preferably a mirror that is closest to you in network terms.

Debian HTTP or FTP mirror addresses can be found at https://www.debian.org/distrib/ftplist (look at the “list of Debian mirrors” section). HTTP mirrors are generally speedier than FTP mirrors.

For example, suppose your closest Debian mirror is **http://mirrors.kernel.org**. When inspecting that mirror with a web browser or FTP program, you will notice that the main directories are organized like this:

```
http://mirrors.kernel.org/debian/dists/jessie/main/binary-i386/...
http://mirrors.kernel.org/debian/dists/jessie/contrib/binary-i386/...
```

To use this mirror with **apt**, you add this line to your **sources.list** file:

```
deb http://mirrors.kernel.org/debian jessie main contrib
```

Note that the “**dists**” is added implicitly, and the arguments after the release name are used to expand the path into multiple directories.

After adding your new sources, disable the previously existing “**deb**” lines in **sources.list** by placing a hash sign (**#**) in front of them.

4.3.2. Adding APT sources for a local mirror

Instead of using HTTP or FTP package mirrors, you may wish to modify **/etc/apt/sources.list** to use a mirror on a local disk (possibly mounted over NFS).

For example, your package mirror may be under **/var/ftp/debian/**, and have main directories like this:

```
/var/ftp/debian/dists/jessie/main/binary-i386/...
/var/ftp/debian/dists/jessie/contrib/binary-i386/...
```

To use this with **apt**, add this line to your **sources.list** file:

```
deb file:/var/ftp/debian jessie main contrib
```

Note that the “**dists**” is added implicitly, and the arguments after the release name are used to expand the path into

multiple directories.

After adding your new sources, disable the previously existing "**deb**" lines in **sources.list** by placing a hash sign (**#**) in front of them.

4.3.3. Adding APT sources from optical media

If you want to use *only* CDs (or DVDs or Blu-ray Discs), comment out the existing "**deb**" lines in **/etc/apt/sources.list** by placing a hash sign (**#**) in front of them.

Make sure there is a line in **/etc/fstab** that enables mounting your CD-ROM drive at the **/media/cdrom** mount point. For example, if **/dev/sr0** is your CD-ROM drive, **/etc/fstab** should contain a line like:

```
/dev/sr0 /media/cdrom auto noauto,ro 0 0
```

Note that there must be *no spaces* between the words **noauto,ro** in the fourth field.

To verify it works, insert a CD and try running

```
# mount /media/cdrom    # this will mount the CD to
the mount point
# ls -alF /media/cdrom  # this should show the CD's
root directory
# umount /media/cdrom   # this will unmount the CD
```

Next, run:

```
# apt-cdrom add
```

for each Debian Binary CD-ROM you have, to add the data about each CD to APT's database.

4.4. Upgrading packages

The recommended way to upgrade from previous Debian releases is to use the package management tool **apt-get**. In previous releases, **aptitude** was recommended for this purpose, but recent versions of **apt-get** provide equivalent functionality and also have shown to more consistently give the desired upgrade results.

Don't forget to mount all needed partitions (notably the root and **/usr** partitions) read-write, with a command like:

```
# mount -o remount,rw /mountpoint
```

Next you should double-check that the APT source entries (in **/etc/apt/sources.list**) refer either to "**jessie**" or to "**stable**". There should not be any sources entries pointing to wheezy.

Note

Source lines for a CD-ROM might sometimes refer to "**unstable**"; although this may be confusing, you should *not* change it.

4.4.1. Recording the session

It is strongly recommended that you use the **/usr/bin/script** program to record a transcript of the upgrade session. Then if a problem occurs, you will have a log of what happened, and if needed, can provide exact information in a bug report. To start the recording, type:

```
# script -t 2>~/upgrade-jessiestep.time -a ~/upgrade-
jessiestep.script
```

or similar. If you have to rerun the typescript (e.g. if you have to reboot the system) use different ***step*** values to indicate which step of the upgrade you are logging. Do not put the typescript file in a temporary directory such as **/tmp** or **/var/tmp** (files in those directories may be deleted during the upgrade or during any restart).

The typescript will also allow you to review information that has scrolled off-screen. If you are at the system's console, just switch to VT2 (using **Alt**+**F2**) and, after logging in, use **less -R ~root/upgrade-jessie.script** to view the file.

After you have completed the upgrade, you can stop **script** by typing **exit** at the prompt.

If you have used the *-t* switch for **script** you can use the **scriptreplay** program to replay the whole session:

```
# scriptreplay ~/upgrade-jessie.time ~/upgrade-
jessie.script
```

4.4.2. Updating the package list

First the list of available packages for the new release needs to be fetched. This is done by executing:

```
# apt-get update
```

4.4.3. Make sure you have sufficient space for the upgrade

You have to make sure before upgrading your system that you will have sufficient hard disk space when you start the full system upgrade described in Section 4.4.5, “Upgrading the system”. First, any package needed for installation that is fetched from the network is stored in **/var/cache/apt/archives** (and the **partial/** subdirectory, during download), so you must make sure you have enough space on the file system partition that holds **/var/** to temporarily download the packages that will be installed in your system. After the download, you will probably need more space in other file system partitions in order to both install upgraded packages (which might contain bigger binaries or more data) and new packages that will be pulled in for the upgrade. If your system does not have sufficient space you might end up with an incomplete upgrade that is difficult to recover from.

apt-get can show you detailed information about the disk space needed for the installation. Before executing the upgrade, you can see this estimate by running:

```
# apt-get -o APT::Get::Trivial-Only=true dist-upgrade
[ ... ]
XXX upgraded, XXX newly installed, XXX to remove and
XXX not upgraded.
Need to get xx.xMB of archives.
After this operation, AAAMB of additional disk space
will be used.
```

Note

Running this command at the beginning of the upgrade process may give an error, for the reasons described in the next sections. In that case you will need to wait until you've done the minimal system upgrade as in Section 4.4.4, “Minimal system upgrade” before running this command to estimate the disk space.

If you do not have enough space for the upgrade, **apt-get** will warn you with a message like this:

```
E: You don't have enough free space in /var/cache
/apt/archives/.
```

In this situation, make sure you free up space beforehand. You can:

- Remove packages that have been previously downloaded for installation (at **/var/cache/apt/archives**). Cleaning up the package cache by running **apt-get clean** will remove all previously downloaded package files.
- Remove forgotten packages. If you have used **aptitude** or **apt-get** to manually install packages in wheezy it will have kept track of those packages you manually installed, and will be able to mark as redundant those packages pulled in by dependencies alone which are no longer needed due to a package being removed. They will not mark for removal packages that you manually installed. To remove automatically installed packages that are no longer used, run:

  ```
  # apt-get autoremove
  ```

  You can also use **deborphan**, **debfoster**, or **cruft** to find redundant packages. Do not blindly remove the packages these tools present, especially if you are using aggressive non-default options that are prone to false positives. It is highly recommended that you manually review the packages suggested for removal (i.e. their contents, sizes, and descriptions) before you remove them.

- Remove packages that take up too much space and are not currently needed (you can always reinstall them after the upgrade). If you have **popularity-contest** installed, you can use **popcon-largest-unused** to list the packages you do not use that occupy the most space. You can find the packages that just take up the most disk space with **dpigs** (available in the **debian-goodies** package) or with **wajig** (running **wajig size**). They can also be found with **aptitude**. Start **aptitude** in "visual mode", select Views → New Flat Package List, press **l** and enter **~i**, then press **S** and enter **~installsize**. This will give you a handy list to work with.

- Remove translations and localization files from the system if they are not needed. You can install the **localepurge** package and configure it so that only a few selected locales are kept in the system. This will reduce the disk space consumed at **/usr/share/locale**.

- Temporarily move to another system, or permanently remove, system logs residing under **/var/log/**.

- Use a temporary **/var/cache/apt/archives**: You can use a temporary cache directory from another filesystem (USB storage device, temporary hard disk, filesystem already in use, ...)

  

  Note

  Do not use an NFS mount as the network connection could be interrupted during the upgrade.

  For example, if you have a USB drive mounted on **/media/usbkey**:

  1. remove the packages that have been previously downloaded for installation:

     ```
     # apt-get clean
     ```

  2. copy the directory **/var/cache/apt/archives** to the USB drive:

```
# cp -ax /var/cache/apt/archives /media
/usbkey/
```

3. mount the temporary cache directory on the current one:

```
# mount --bind /media/usbkey/archives
/var/cache/apt/archives
```

4. after the upgrade, restore the original **/var/cache/apt/archives** directory:

```
# umount /media/usbkey/archives
```

5. remove the remaining **/media/usbkey/archives**.

You can create the temporary cache directory on whatever filesystem is mounted on your system.

- Do a minimal upgrade of the system (see Section 4.4.4, “Minimal system upgrade”) or partial upgrades of the system followed by a full upgrade. This will make it possible to upgrade the system partially, and allow you to clean the package cache before the full upgrade.

Note that in order to safely remove packages, it is advisable to switch your **sources.list** back to wheezy as described in Section A.2, “Checking your sources list”.

4.4.4. Minimal system upgrade

In some cases, doing the full upgrade (as described below) directly might remove large numbers of packages that you will want to keep. We therefore recommend a two-part upgrade process: first a minimal upgrade to overcome these conflicts, then a full upgrade as described in Section 4.4.5, “Upgrading the system”.

To do this, first run:

```
# apt-get upgrade
```

Note

The upgrade process for some previous releases recommended the use of **aptitude** for the upgrade. This tool is not recommended for upgrades from wheezy to jessie.

This has the effect of upgrading those packages which can be upgraded without requiring any other packages to be removed or installed.

The minimal system upgrade can also be useful when the system is tight on space and a full upgrade cannot be run due to space constraints.

If the **apt-listchanges** package is installed, it will (in its default configuration) show important information about upgraded packages in a pager. Press **q** after reading to exit the pager and continue the upgrade.

4.4.5. Upgrading the system

Once you have taken the previous steps, you are now ready to continue with the main part of the upgrade. Execute:

```
# apt-get dist-upgrade
```

Note

The upgrade process for some previous releases recommended the use of **aptitude** for the upgrade. This tool is not recommended for upgrades from wheezy to jessie.

This will perform a complete upgrade of the system, installing the newest available versions of all packages, and resolving all possible dependency changes between packages in different releases. If necessary, it will install some new packages (usually new library versions, or renamed packages), and remove any conflicting obsoleted packages.

When upgrading from a set of CD-ROMs (or DVDs), you will be asked to insert specific CDs at several points during the upgrade. You might have to insert the same CD multiple

times; this is due to inter-related packages that have been spread out over the CDs.

New versions of currently installed packages that cannot be upgraded without changing the install status of another package will be left at their current version (displayed as “held back”). This can be resolved by either using **aptitude** to choose these packages for installation or by trying **apt-get install *package***.

4.5. Possible issues during upgrade

The following sections describe known issues that might appear during an upgrade to jessie.

4.5.1. Dist-upgrade fails with “Could not perform immediate configuration”

In some cases the **apt-get dist-upgrade** step can fail after downloading packages with:

```
E: Could not perform immediate configuration on
'package'.  Please see man 5 apt.conf under
APT::Immediate-Configure for details.
```

If that happens, running **apt-get dist-upgrade -o APT::Immediate-Configure=0** instead should allow the upgrade to proceed.

Another possible workaround for this problem is to temporarily add both wheezy and jessie sources to your **sources.list** and run **apt-get update**.

4.5.2. Expected removals

The upgrade process to jessie might ask for the removal of packages on the system. The precise list of packages will vary depending on the set of packages that you have installed. These release notes give general advice on these removals, but if in doubt, it is recommended that you examine the package removals proposed by each method before proceeding. For more information about packages obsoleted in jessie, see Section 4.10, “Obsolete packages”.

4.5.3. Conflicts or Pre-Depends loops

Sometimes it's necessary to enable the **APT::Force-LoopBreak** option in APT to be able to temporarily remove an essential package due to a Conflicts/Pre-Depends loop. **apt-get** will alert you of this and abort the upgrade. You can work around this by specifying the option **-o APT::Force-LoopBreak=1** on the **apt-get** command line.

It is possible that a system's dependency structure can be so corrupt as to require manual intervention. Usually this means using **apt-get** or

```
# dpkg --remove package_name
```

to eliminate some of the offending packages, or

```
# apt-get -f install
# dpkg --configure --pending
```

In extreme cases you might have to force re-installation with a command like

```
# dpkg --install /path/to/package_name.deb
```

4.5.4. File conflicts

File conflicts should not occur if you upgrade from a “pure” wheezy system, but can occur if you have unofficial backports installed. A file conflict will result in an error like:

```
Unpacking <package-foo> (from <package-foo-file>) ...
dpkg: error processing <package-foo> (--install):
 trying to overwrite `<some-file-name>',
 which is also in package <package-bar>
dpkg-deb: subprocess paste killed by signal (Broken
pipe)
 Errors were encountered while processing:
 <package-foo>
```

You can try to solve a file conflict by forcibly removing the package mentioned on the *last* line of the error message:

```
# dpkg -r --force-depends package_name
```

After fixing things up, you should be able to resume the

upgrade by repeating the previously described **apt-get** commands.

4.5.5. Configuration changes

During the upgrade, you will be asked questions regarding the configuration or re-configuration of several packages. When you are asked if any file in the **/etc/init.d** directory, or the **/etc/manpath.config** file should be replaced by the package maintainer's version, it's usually necessary to answer “yes” to ensure system consistency. You can always revert to the old versions, since they will be saved with a **.dpkg-old** extension.

If you're not sure what to do, write down the name of the package or file and sort things out at a later time. You can search in the typescript file to review the information that was on the screen during the upgrade.

4.5.6. Change of session to console

If you are running the upgrade using the system's local console you might find that at some points during the upgrade the console is shifted over to a different view and you lose visibility of the upgrade process. For example, this may happen in desktop systems when the display manager is restarted.

To recover the console where the upgrade was running you will have to use **Ctrl**+**Alt**+**F1** (if in the graphical startup screen) or **Alt**+**F1** (if in the local text-mode console) to switch back to the virtual terminal 1. Replace F1 with the function key with the same number as the virtual terminal the upgrade was running in. You can also use **Alt**+**Left Arrow** or **Alt**+**Right Arrow** to switch between the different text-mode terminals.

4.5.7. Special care for specific packages

In most cases, packages should upgrade smoothly between wheezy and jessie. There are a small number of cases where some intervention may be required, either before or during the upgrade; these are detailed below on a per-package basis.

4.5.7.1. systemd

The Debian upgrade from wheezy to jessie will by default

migrate your init system from the SysV to systemd. Depending on your system and setup, you may need to do some manual changes. We have detailed known issues in Section 5.6, “Upgrading installs the new default init system for Jessie”.

4.5.7.2. LXC

If you have LXC installed, you may need special care when upgrading your system and your containers. Please have a look at Section 5.8, “Upgrade considerations for LXC hosts and containers” for known issues and solutions.

4.6. Upgrading your kernel and related packages

This section explains how to upgrade your kernel and identifies potential issues related to this upgrade. You can either install one of the **linux-image-*** packages provided by Debian, or compile a customized kernel from source.

Note that a lot of information in this section is based on the assumption that you will be using one of the modular Debian kernels, together with **initramfs-tools** and **udev**. If you choose to use a custom kernel that does not require an initrd or if you use a different initrd generator, some of the information may not be relevant for you.

4.6.1. Installing a kernel metapackage

When you dist-upgrade from wheezy to jessie, it is strongly recommended that you install a linux-image-* metapackage, if you have not done so before. These metapackages will automatically pull in a newer version of the kernel during upgrades. You can verify whether you have one installed by running:

```
# dpkg -l "linux-image*" | grep ^ii | grep -i meta
```

If you do not see any output, then you will either need to install a new linux-image package by hand or install a linux-image metapackage. To see a list of available linux-image metapackages, run:

```
# apt-cache search linux-image- | grep -i meta |
grep -v transition
```

If you are unsure about which package to select, run **uname -r** and look for a package with a similar name. For example, if you see '**2.6.32-5-amd64**', it is recommended that you install **linux-image-amd64**. You may also use **apt-cache** to see a long description of each package in order to help choose the best one available. For example:

```
# apt-cache show linux-image-amd64
```

You should then use **apt-get install** to install it. Once this new kernel is installed you should reboot at the next available opportunity to get the benefits provided by the new kernel version. However, please have a look at Section 4.7, “Things to do before rebooting” before performing the first reboot after the upgrade.

For the more adventurous there is an easy way to compile your own custom kernel on Debian. Install the kernel sources, provided in the **linux-source** package. You can make use of the **deb-pkg** target available in the sources' makefile for building a binary package. More information can be found in the Debian Linux Kernel Handbook, which can also be found as the **debian-kernel-handbook** package.

If possible, it is to your advantage to upgrade the kernel package separately from the main **dist-upgrade** to reduce the chances of a temporarily non-bootable system. Note that this should only be done after the minimal upgrade process described in Section 4.4.4, “Minimal system upgrade”.

4.6.2. Changes to root and /usr filesystem mounting and checking

initramfs-tools will now also run **fsck** on the root filesystem before mounting it. If the chosen init program is **systemd** and there is a separate **/usr** filesystem, it will also fsck and mount **/usr**.

- If **/usr** is a separate filesystem on a RAID device and the **INITRDSTART** setting in **/etc/default/mdadm** is not '**all**', you will need to change it to include that device.

- If **/usr** is a separate filesystem on an LVM logical volume, and the line for **/usr** in **/etc/fstab** specifies the device by **UUID** or **LABEL**, you must change this line to specify the device using the format ***/dev/mapper/VG-LV*** or ***/dev/VG/LV***.

- It is no longer possible to bind-mount the **/usr** filesystem.

- If the RTC (real time clock) is set to local time and the local time is ahead of UTC, **e2fsck** will print a warning during boot about the time changing backward (bug #767040). You can disable this by putting the following lines in **/etc/e2fsck.conf**:

```
[options]
broken_system_clock=1
```

4.7. Things to do before rebooting

When **apt-get dist-upgrade** has finished, the "formal" upgrade is complete, but there are some other things that should be taken care of *before* the next reboot.

- When upgrading from Wheezy to Jessie, it can be a good idea to purge old packages *before* the first reboot. In particular, obsolete init scripts may cause issues. Please see Section 4.8.1, "Purging removed packages" for details on finding and purging removed packages.

4.8. Preparing for the next release

After the upgrade there are several things you can do to prepare for the next release.

- Remove newly redundant or obsolete packages as described in Section 4.4.3, "Make sure you have sufficient space for the upgrade" and Section 4.10, "Obsolete packages". You should review which configuration files they use and consider purging the packages to remove their configuration files. See also Section 4.8.1, "Purging removed packages".

4.8.1. Purging removed packages

It is generally advisable to purge removed packages. This is

especially true if these have been removed in an earlier release upgrade (e.g. from the upgrade to wheezy) or they were provided by third-party vendors. In particular, old init.d scripts have been known to cause issues.

Caution

Purging a package will generally also purge its log files, so you might want to back them up first.

The following command displays a list of all removed packages that may have configuration files left on the system (if any):

```
# dpkg -l | awk '/^rc/ { print $2 }'
```

The packages can be removed by using **apt-get purge**. Assuming you want to purge all of them in one go, you can use the following command:

```
# apt-get purge $(dpkg -l | awk '/^rc/ { print $2 }')
```

If you use **aptitude**, you can also use the following alternative to the commands above:

```
$ aptitude search '~c'
$ aptitude purge '~c'
```

4.9. Deprecated components

With the next release of Debian 9 (codenamed stretch) some features will be deprecated. Users will need to migrate to other alternatives to prevent trouble when updating to 9.

This includes the following features:

- The **hardening-wrapper** package is deprecated and is expected to be removed in Stretch.

4.10. Obsolete packages

Introducing lot of new packages, jessie also retires and omits quite some old packages that were in wheezy. It provides no upgrade path for these obsolete packages. While nothing

prevents you from continuing to use an obsolete package where desired, the Debian project will usually discontinue security support for it a year after jessie's release[5], and will not normally provide other support in the meantime. Replacing them with available alternatives, if any, is recommended.

There are many reasons why packages might have been removed from the distribution: they are no longer maintained upstream; there is no longer a Debian Developer interested in maintaining the packages; the functionality they provide has been superseded by different software (or a new version); or they are no longer considered suitable for jessie due to bugs in them. In the latter case, packages might still be present in the “unstable” distribution.

Detecting which packages in an updated system are “obsolete” is easy since the package management front-ends will mark them as such. If you are using **aptitude**, you will see a listing of these packages in the “Obsolete and Locally Created Packages” entry.

The Debian Bug Tracking System often provides additional information on why the package was removed. You should review both the archived bug reports for the package itself and the archived bug reports for the ftp.debian.org pseudo-package.

The list of obsolete packages includes:

- **postgresql-9.1**, successor is **postgresql-9.4**. Once the operating system upgrade is finished, you should plan to also upgrade your PostgreSQL 9.1 database clusters to the new PostgreSQL version 9.4 using the **pg_upgradecluster** tool. For users of the PL/perl procedural language, jessie provides an updated **postgresql-plperl-9.1** package linked against jessie's version of libperl in order to enable upgrading to the new perl version in jessie while keeping the old PL/perl database functions usable until the database is upgraded as well.

- **python3.2**, successor is **python3.4**. (Version 2.7 is supported in both wheezy and jessie.)

- **ruby1.8** and **ruby1.9.1**; successor is **ruby2.1**. Please install the package **ruby** to automatically track the

current ruby version.

- **mplayer**; alternatives are **mplayer2**, and **mpv** (new in jessie). Whilst the former is mostly compatible with mplayer in terms of command-line arguments and configuration (and adds a few new features too), the latter adds a lot of new features and improvements, and it is actively maintained upstream.
- **openoffice.org**; please use **libreoffice**.
- **squid**, successor is **squid3**.
- **libjpeg-progs**, successor is **libjpeg-turbo-progs**.
- **openjdk-6-***, successor is **openjdk-7-***.

4.10.1. Dummy packages

Some packages from wheezy have been split into several packages in jessie, often to improve system maintainability. To ease the upgrade path in such cases, jessie often provides “dummy” packages: empty packages that have the same name as the old package in wheezy with dependencies that cause the new packages to be installed. These “dummy” packages are considered redundant after the upgrade and can be safely removed.

Most (but not all) dummy packages' descriptions indicate their purpose. Package descriptions for dummy packages are not uniform, however, so you might also find **deborphan** with the **--guess-*** options (e.g. **--guess-dummy**) useful to detect them in your system. Note that some dummy packages are not intended to be removed after an upgrade but are, instead, used to keep track of the current available version of a program over time.

[1] If the debconf priority is set to a very high level you might prevent configuration prompts, but services that rely on default answers that are not applicable to your system will fail to start.

[2] For example: DNS or DHCP services, especially when there is no redundancy or failover. In the DHCP case end-users might be disconnected from the network if the lease time is

lower than the time it takes for the upgrade process to complete.

[3] This feature can be disabled by adding the parameter **`panic=0`** to your boot parameters.

[4] Debian's package management system normally does not allow a package to remove or replace a file owned by another package unless it has been defined to replace that package.

[5] Or for as long as there is not another release in that time frame. Typically only two stable releases are supported at any given time.


Chapter 3. Installation System


Chapter 5. Issues to be aware of for jessie