DOCUMENT MOVED TO DISCOURSE. NO FURTHER EDITS HERE!

https://discourse.ubuntu.com/t/technical-board-feedback-requested-draft-policy-on-third-party-software-sources-included-by-ubuntu/46849

The text prior to the move is preserved here for reference of editing history and comments.

Third party repository requirements

Background

This document is for use by Technical Board members to come up with a set of requirements for packages in Ubuntu that enable third party software repositories to be installed by default.

See: https://lists.ubuntu.com/archives/technical-board/2021-June/002559.html and https://irclogs.ubuntu.com/2021/07/13/%23ubuntu-meeting.html#t18:59

See also: https://wiki.ubuntu.com/UbuntuSeededSnaps

Draft status [2022-06-21]: the TB seems to have reached a rough consensus on most topics, so rbasak has started drafting the actual wording for a final policy document. He has tried to incorporate all views expressed by TB members and further discussion, but how well this actually represents all TB member views is subject to further review.

How to read this document

- The main part of this document defines the Purpose, Scope and specifies General Requirements, the mechanism for Exceptions, and the Specific Principles that form the essence of this policy.
- Appendix A: General Exceptions documents exceptions that we make to our principles in different classes of special cases.
- Appendix B: Snap Specifics documents how we achieve compliance in the specific case of Snaps, including Snap-specific interpretations, exceptions to the general policy both for Snaps in general and specific Snap packages, to be ratified by the Technical Board.
- Appendix C: Matters not yet addressed documents outstanding issues, such as current known instances of lack of compliance, any plans we have on how to achieve compliance in the future, and future requirements that we think we may wish to move towards, but to which we have not yet committed. However, to make progress, our

intention is to push ahead with applying the policy to future changes, and deal with the backlog of unresolved issues over time.

Plan of Attack

- (Done) Consult with TB members and draft and adjust until consensus is reached
- (Done) Consult with stakeholders from whom implementation will be required (eg. Snap Store).
- (Done) Consult with affected flavours
- Consult with snap related stakeholders within Canonical to ensure that there are no hard blockers to implementing what this policy would require once finalised. Note that this is approval for an in-progress draft. **By approving you're confirming that you agree this is the way forward and you don't consider there to be any major blockers to implementation from your perspective if these were to be approved.** But we might still be tweaking things based on feedback from others (including community).

Approver	Approval Status	Approval Date
Oliver Smith	Approved	
Samuele Pedroni	Under review	
Alfonso Sanchez-B…	Not started	
Philip Meulengracht	Not started	
Sebastien Bacher	Approved	3 Jun 2024

- Consult with Ubuntu developers at large
- Ratification by the Technical Board
  - To make progress, initially the Technical Board will ratify only the main body of the document and exclude the appendices.
- Document somewhere more permanent and public

Discourse Post

Robie will move this draft to Discourse to gather further feedback from Ubuntu Developers and the wider community. The post will explain the plan as well as present the current state of the draft. After consultation and any adjustments, the aim is for the Technical Board to ratify the draft into a formal policy position, which will then be expected to be binding on Ubuntu developers making further changes to the Ubuntu (deb) archive and on any snap publishers that are brought in through that mechanism.

The introduction on Discourse post will be Robie's words (so do not need TB approval as such) and are drafted as follows. Feedback appreciated!

Subject: [Technical Board] Feedback requested: draft policy on third party software sources included by Ubuntu

When a user installs Ubuntu LTS, they expect that the platform and default apps, together with apps they install from the default repositories, will follow some principles of quality and stability. For example, they generally expect that behaviour won't change in surprising ways for its 12-year life, that there will be some form of support for it in that stable state for that long, and that Ubuntu Developers are able to maintain it according to these expectations. We have also always maintained our non-LTS (“interim”) releases to the same stability standards.

Now that we have snaps installed by default and a growing series of "snap installer" debs being added to the deb archive, what does this mean for these expectations? Snaps are applicable much more widely than just the software that Ubuntu ships itself. Specific snaps and snap tracks *may* match our standards for "LTS"-ness, or they may not. In general, proprietary snaps would presumably be unacceptable for us to ship by default, but we should have that defined somewhere. Their licences may change on a new version. And so on.

There are exceptions. Probably the most notable one is Firefox, where we do accept changes in behaviour in the latest version into our stable releases. Users can also "opt in" to the upgrade treadmill for anything else by installing the snap for it. In that case, changes in behaviour wouldn’t be surprising because they opted in.

There are also properties that existed in the traditional model but do not necessarily exist in the snap world that we would like to keep. For example, a foundational property of our ecosystem is that users have the ability to download sources, modify and patch their own systems, as well as help other users receive the benefit of these abilities, even if the software publishers choose to go in a different direction. We’d like to maintain that ability for anything that appears in Ubuntu by default.

This isn't just a snap-specific question. The same sort of questions have previously been raised for Flatpaks.

When this was raised to the Technical Board, we formed a view that we should have some written policy that defines what users can expect from snaps that are installed in this way, together with a process for granting exceptions. Since we are working on this retrospectively, it will be the case that some existing packaging does not comply with our principles. In time, for each outlier we intend to figure out if we should make exceptions, bring those packages into line, or something in the middle. We are starting by defining where we want to be, then we can apply that standard to new cases, and then we can work towards.

Our current draft follows. Feedback is appreciated.

Purpose

Ubuntu maintains its own standards for certain aspects of quality and behaviour in relation to software that it ships, with the goal that users can expect and rely upon this. For example, the "Ubuntu LTS" label comes with an expectation that we do not generally disrupt users with free-for-all improvements during the lifetime of such an LTS release.

With an increasing reliance on snaps to ship to Ubuntu users by default, it's important to continue setting appropriate expectations for quality and behaviour outside the traditional "deb" archive and ensuring that they are met. For example, snaps are intended to delegate quality considerations and decisions to the snap publisher: anyone can publish snaps. But we want the snaps that Ubuntu ships by default, or otherwise makes available by default, to meet our own minimum quality standards.

This document is intended to set a baseline for these expectations in general—for debs, snaps and any other packaging system that might be used in the future.

Scope

This document specifies Ubuntu's baseline policy requirements on all software present in a default software source. This includes software sources that are enabled automatically upon the installation of some other software present in a repository that is enabled by default. Examples:

1. A snap may be installed on a default Ubuntu installation.
2. Installing a deb from the Ubuntu archive may result in the pulling in of a snap to support an upgrade path.
3. Someone may want the installation of a deb from the Ubuntu archive to automatically enable and pull in a Flatpak from FlatHub.
4. The "Software Centre" of an Ubuntu flavour may choose to enable, by default, a third party package repository of some form.

Any behaviour of this kind must comply with the requirements specified in this document.

The requirements documented here are intended to fulfil user expectations and similar issues that arise in relation to software origin. These are not the only set of requirements that exist. For example, Debian policy and the derived Ubuntu policy still apply for debs.

Not in scope

If all routes to install some other software involves users explicitly choosing to opt in to some other software source and such that they are reasonably informed or is otherwise reasonably expected to understand that this choice means that Ubuntu is not responsible for software installed from this source, then this policy does not apply.

General Requirements

Review and approvals

Use of any new class of packaging system to make software appear or available by default on Ubuntu must receive explicit approval from the Technical Board: compliance with this document is an expectation but not sufficient in itself.

For example, the Technical Board expects to decline to approve PPAs as a mechanism that may be enabled in Ubuntu by default. Use of PPAs would harm auditability, diffuse community oversight, and bypass the release process rather than integrate with it. The Ubuntu archive should be used instead.

Exceptions

Various packages are exceptional for various reasons, and Ubuntu has a tradition of being pragmatic about this in order to deliver the best experience to our users. This document merely specifies the baseline. Exceptions will continue to be made on a case-by-case basis by the Technical Board or by any other teams as delegated by them. Exceptions granted by the Technical Board are documented in Appendix A: General Exceptions.

Specific Principles

Principle 1: behaviour will remain "stable" for the lifetime of an Ubuntu release.

The package publisher must agree that the package and any subsequent updates presented to Ubuntu users by default will remain stable for the lifetime of the Ubuntu release.

"Stable" means that behaviour from users' perspectives will not change, except to fix bugs, unless the change already matches a category of standing exception generally granted for Ubuntu packages themselves, as defined at https://wiki.ubuntu.com/StableReleaseUpdates#When under the "When" section.

Since updates may be made to the package without review by Ubuntu, this requirement is enforced by trust only. Ubuntu would appreciate consultation to discuss where we stand on individual issues that fall close to the line. Ultimately, if Ubuntu is unhappy about the way this requirement is being met, then the package, and other packages from the same maintenance teams, may be removed from publication by default in future Ubuntu releases to help meet the user expectation of stable release stability.

Rationale: Ubuntu users expect, for most packages, that a stable Ubuntu release remains stable in the sense that functionality and behaviour does not change until they opt-in to upgrading to a subsequent release. There are a number of pragmatic exceptions to this rule; the list of these should continue to be managed under the governance of the Technical Board.

Principle 2: Ubuntu developers can override and patch packages

There must be a mechanism that can be used by Ubuntu developers to override and patch the package at source level, such that the updated package gets delivered to users without any further intervention from their perspective apart from the usual update process.

Outside debs shipped through the primary archive, it is not expected that this mechanism will be invoked, but the mechanism must exist for use by Ubuntu developers as a last resort.

Some modifications may not be permitted due to law, such as trademark law. Such modifications are out of scope of this requirement.

A documented test plan must be maintained that is used to validate a package build before publication for general use. This may be manual or automatic; if automatic then the automation must be reproducible for use by Ubuntu developers.

Rationale: if a package is found not to meet Ubuntu's quality standards, it must be possible for Ubuntu developers to be able to fix it in order to uphold these standards. This already exists for the Ubuntu deb archive, and should continue to exist for anything Ubuntu ships by default. Issues related to law remain out of scope and their effects unchanged.

Principle 3: package maintainer agrees to maintain packages for the lifetime of the Ubuntu release.

Maintainers of the package must agree to maintain the package for the lifetime of each Ubuntu release in which the package was made available.

This means that security and bug fix updates that are consistent with the other requirements must be made available to the user of the package in the normal way.

Since updates may be made to the package without review by Ubuntu, this requirement, and the precise meaning of "maintain" such as what fixes would qualify, is enforced by trust only. Ultimately, if Ubuntu is unhappy about the way this requirement is being met, then the package, and other packages from the same maintenance teams, may be removed from publication by default in future Ubuntu releases to help meet the user expectation of stable release stability.

Rationale: this is an expectation that exists for the Ubuntu deb archive, and should continue to exist for anything Ubuntu ships by default.

Principle 4: licensing would be acceptable to Ubuntu ie. dfsg-free

The package must be licensed under terms acceptable under the Debian Free Software Guidelines.

Rationale: these are the same requirements Ubuntu has always applied to its main archive.

Principle 5: packages are built on a build farm that is trusted by the Ubuntu project

Packages must be built by and published from Ubuntu’s own infrastructure (ie. Launchpad and the Snap Store).

Rationale: user security depends on trustworthiness of the build and publication process and is best served by keeping the number of organisations that must be trusted to a minimum; namely, one.

Principle 6: sources for published builds are retained and publicly available

Sources and build logs for published builds must be retained, be publicly available, and the particular source tree and build log discoverable by a user who has a corresponding binary package installed.

Rationale: this empowers users to modify their installed packages themselves. This is the essence of Free Software and is a long-standing expectation of users of Ubuntu.

Principle 7: available on all architectures supported by Ubuntu

Packages must be available on all architectures supported by Ubuntu. If you cannot support an architecture, please ask the TB for an exception.

Rationale: the expectation of a user of a supported architecture is that the user experience is not degraded in relation to any other supported architecture.

Principle 8: public bug trackers are available

It must be possible for a user to locate and post in a public bug tracker about an issue they are facing with an installed package.

Rationale: an inability to do this harms the ability for users to develop and share modifications to packages. Such modifications are the essence of Free Software and the ability to communicate with other users in a bug database is a long-standing expectation of users of Ubuntu.

Appendix A: General Exceptions

Exception to Principle 7: available on all architectures supported by Ubuntu

If Launchpad does not allow the general public to build on a particular architecture, then we grant an exception that building on that architecture is not required. We strongly recommend that all architectures remain enabled; in that case it is fine that if builds or tests fail then automation prevents publication on the unavailable architectures. This allows future fixes to architecture support to land without further intervention.

Desktop packages

This section only applies to a package that, regardless of architecture support, could only practically be used exclusively by desktop users.

If such a package is only used for the purposes of being made available in some flavours (by seeding or presenting it for user installation in a flavour-specific way), then only the architectures supported by those flavours need to be supported.

This exception may exclude the opportunity for those architectures from doing “remote desktop hosting” because widespread support across packages for those architectures may be missing as a result. We think this is an acceptable trade-off between this possibility and the burden of maintaining such packages in those architectures.

Specific packages

If it is not practical to maintain full architecture support for some specific package, and the more general exceptions above cannot be used, then the Technical Board may grant and document an exception here. The following packages currently have such exceptions:

- Firefox. Must build on amd64, arm64 and armhf. Reason: unsupported upstream, too complex to fix at the distro level.
- snap-store (ie. the new one currently on the preview channel).
- Ubuntu desktop installer. Uses Flutter for which there isn’t a runtime on other arches.

Appendix B: Snap Specifics

How Snaps Implement the Principles

Track mappings and ICE branches

Principles 1 and 2 are implemented in snaps through the use of *track mappings* and *ICE branches* as defined here. Tracks and branches are aspects of snap channels.

- **Track mappings.** To maintain stable behaviour for the lifetime of a given Ubuntu release, that release is mapped to a snap track for each snap that is within scope. For example, we might declare that snap package “foo” shall, by default on a fresh installation of Ubuntu 24.04, be installed and mapped to track “15” of the “foo” snap package. This mapping would be expected to remain fixed for the lifetime of Ubuntu 24.04.
- **ICE branches.** snapd may be configured to follow a given channel branch for a particular installed snap, and this shall be done for all snaps when their installation falls under the scope of this policy. The branch name shall be “ubuntu-XX.YY”, where XX.YY is the Ubuntu release version in question. For example, on Ubuntu 24.04, the firefox snap should track “latest/stable/ubuntu-24.04” by default. For the purposes of this policy, we’ll call this snap branch the in-case-of emergency branch (“ICE branch”). Since the Firefox snap holds an exception to principle 1, only principle 2 is implemented in this example.

When a user upgrades to a new Ubuntu distribution release, the track mapping is often expected to change. As an exception, the track mapping may also change during the lifetime of a stable release.

Behaviour on release upgrades

1. **Snaps installed by default.** If a snap was installed by default on an Ubuntu installation, it is preferable for users to automatically refresh to the new default snap channel automatically (including track, risk and branch) regardless of whether they

have previously deviated from previous default ICE-enabled channel. This practice is consistent with other package types and PPAs where release upgrades aim to create a consistent 'standard' configuration. For example: on upgrade from 24.04 to 26.04, the Software Centre snap should be refreshed to the track mapping and stable risk channel for 26.04 regardless of whether the user had adjusted it before.

2. **Snaps otherwise within scope.** If a snap was not installed by default, but instead installed by a different path that nevertheless falls within the scope of this document, then it should be refreshed to the new track mapping only if the user had not deviated from it tracking the ICE branch. For example, the lxd transitional deb on 24.10 might arrange for the snap to track 5.20/stable/ubuntu-24.10 based on its track mapping for 24.10. If on upgrade to 25.04 it is still tracking a branch that matches /^ubuntu-/ and the track mapping has changed to 5.21, then it would be migrated to track 5.21/stable/ubuntu-25.04 instead. However, had the user deviated away from tracking the ICE branch, that case would be indistinguishable from the following case "Snaps not in scope" and behaviour shall be as if the snap were no longer in scope. Therefore, users can opt a snap out of falling under this case by bringing them out of scope using `snap refresh` or `snap switch` to no longer track the ICE branch.
3. **Snaps not in scope.** If the snap is not installed by default on an Ubuntu installation and was installed by the user manually, then the channel it tracks should not change. This case is detected by the previous channel being tracked not having an ICE branch included.

Track mapping changes during release upgrades shall be handled between ubuntu-release-upgrader and transitional debs as required. If the snap being migrated was previously not tracking an ICE branch, the user should be notified so that they can manually return to the previous state if preferred.

Exceptional track mapping changes during the lifetime of an Ubuntu release may occur. These would require approval from the Technical Board on a case-by-case basis. They would be expected to be handled via the SRU of a suitable deb package which would then implement this logic in its postinst. We do not consider it necessary to implement this unless and until such an exception becomes necessary.

ICE branch details

Normally, ICE branches are tracked by the snapd client but nevertheless do not exist, and in this case snapd falls back to not using any branch. For example, while tracking "latest/stable/ubuntu-24.04" it would install and maintain Firefox from the "latest/stable" channel so long as "latest/stable/ubuntu-24.04" does not exist. However, it does still continue to track "latest/stable/ubuntu-24.04" even if it has applied this fallback mechanism, so future publication into the ICE branch would cause users to switch to this published build, enabling principle 2.

After a release upgrade, the name of an ICE branch changes. If "Behaviour on release upgrades" above requires the snap to be migrated, then the ICE branch name being tracked should also change as part of the upgrade.

To identify "Snaps otherwise within scope", for robustness, we shall consider the tracking of any ICE branch name as a qualifier, not just the specific ICE branch matching the release from which we are upgrading. For example, if the system is tracking "latest/beta/ubuntu-18.04" at the time of upgrading from Ubuntu 22.04 to Ubuntu 24.04, the upgrade process would change the channel being tracked to “15/stable/ubuntu-24.04” if that’s what “Behaviour on release upgrades” says, even though the previous track didn’t match the previous mapping, and even though the previous ICE branch name didn’t match the release from which we were upgrading.

Summary

The specifics are documented above, but to avoid any doubt, the effect of the snap implementations of principles 1 and 2 are that snaps installed that are in scope shall usually combine to track a channel where the track is specified by the track mapping, the risk is `stable` and the branch is `ubuntu-XX.YY` where `XX.YY` is the Ubuntu release of the installed system. Exceptionally, if the Technical Board has granted an exception from principle 1 (eg. Firefox) then the track would be `latest` instead. For example, if Libreoffice were to ship as a snap and did not have an exception against principle 1 then a default 24.04 desktop system might track `24/stable/ubuntu-24.04`. It would be required that there is a `24` track in this example since use of a `latest` track would violate principle 1. Firefox on the same system would be expected to track `latest/stable/ubuntu-24.04` since it does have an exception against principle 1.

Principle 1: behaviour will remain "stable" for the lifetime of an Ubuntu release

This is achieved by use of track mappings combined with established behaviour by the snapd client. The track mapping of a snap that is within scope must use a track for which upstream commits to maintain this principle. This means that `latest` is not a permitted track mapping unless an exception exists as below.

We then rely on the snap maintainer to keep the track stable. This is the normal expectation anyway for snap tracks that aren’t `latest`; we shall rely on trust as is permitted by the requirement.

Exceptions

If the Technical Board has granted an exception from this requirement, then we may map to the “latest” track instead. For example, Firefox has such an exception and maps to the “latest” snap track.

If the Technical Board were to grant an exception after release, then the mapping will have changed after release. This might happen for the same reason that we might apply to a deb package under the existing security and SRU processes, such as if it is necessary to keep users secure, or because the previously mapped track has become unusable due to changes in our ecosystem outside Ubuntu.

Principle 2: Ubuntu developers will be able to override and patch the package

This is achieved by use of ICE branches combined with established behaviour by the snapd client.

This requires the Technical Board to have the ultimate authority to publish into such an ICE branch. It is a condition of the permission to include snaps into Ubuntu that the Snap Store admins defer to the Technical Board's authority and permit such a publication should, in the sole opinion of the Technical Board, the need arises.

Principle 3: the package maintainer agrees to maintain the package for the lifetime of the Ubuntu release

We will document explicit agreement from individual snap maintainers.

Principle 4: licensing would be acceptable to Ubuntu ie. dfsg-free

This will be managed to the same standard as Ubuntu's deb archive. At a minimum, an initial check is required by a competent person before moving a snap within scope, and a process by which deviations can be reported and resolved.

Principle 5: package is built on a build farm that is trusted by the Ubuntu project

Snaps can be built on Launchpad, or may be built by third parties and then uploaded. It might technically be possible to enforce that particular snaps are built on Launchpad, but that would be a feature request that would need to go to the Snap Store. In the meantime, we should ensure that snaps within scope are all being built on Launchpad, and get a commitment from publishers that they will continue to do so.

Principle 6: sources for published builds are retained and publicly available

This is not currently the case, although the required metadata may be available internally. Implementation will be required in Launchpad/Snap Store.

Principle 7: published on all architectures supported by Ubuntu

This should be straightforward with principle 5 implemented, except where some dependency isn't available on some architecture. Compliance will be checked for each in-scope snap once, and a bug can be filed for identified issues in the future.

Principle 8: a public issue/bug tracker is available

Compliance will be checked for each in-scope snap once. A bug can be filed against the general "Ubuntu" project in Launchpad for identified issues in the future, and this can be assigned to the appropriate team.

Additional requirements

The TB agreed in their meeting on 30 May 2023 that any new seeded snaps must be announced in advance to ubuntu-devel@, to help the TB monitor the quality of seeded snaps whose maintainers may not necessarily be Ubuntu developers.

Appendix C: Matters not yet addressed

Changes agreed in principle, but yet to be executed

Snap Store and ICE branches

Currently there are snaps that fall within scope where ICE branches aren't being used, or where they are used the TB doesn't have immediate permission to publish to them. In discussions with the Snap Store team we agreed that this should be possible to arrange in principle, but this has yet to be arranged technically. However, in the meantime, we can ask the Snap Store admins for exceptional access should it be required. Since we expect the need for access to be an exceptionally rare event, it does not seem appropriate to block progress waiting on a technical ACL implementation.

Snaps yet to be audited

We need to ensure that each snap within scope is either within compliance of the principles or that an appropriate exception is granted and documented. That work is yet to be completed. Here is the current status of the snaps identified to be within scope so far:

Existing snaps that are in scope

The following snaps are identified to fall under the scope of these requirements:

Via Ubuntu-specific deb packages

- chromium
- firefox
- lxd (also seeded)
- snapcraft

Via deb package as synced from Debian:

- cyphesis-cpp
- ember

Via seeds

- snapd itself (Samuele says: *snapd itself is also a snap that we ship in images, it uses latest/stable like firefox and the deb has a SRU-exception(?). The snapd deb also installs it if not already present when installing snaps*)
- freeshow (Ubuntu Studio)
- Some of gnome-3-{28,34,38}-{18,20}04
- gtk-common-themes
- Only in Bionic:
    - gnome-calculator

    - gnome-characters
    - gnome-log
    - gnome-system-monitor
    - pulsemixer (MATE)
- snapd-desktop-integration
- software-boutique (MATE)
- ubuntu-budgie-welcome (Budgie)
- ubuntu-mate-welcome (MATE)
- lxd (also through transitional deb)
- snap-store
- ubuntu-desktop-installer/classic=latest/edge (from wsl)

Via installer image generation:

- subiquity

Principle 3: the package maintainer agrees to maintain the package for the lifetime of the Ubuntu release

We will arrange a place where agreements made can be documented and complete that for each snap within scope.

Principle 4: licensing would be acceptable to Ubuntu ie. dfsg-free

- Individual analysis pending: chromium, cyphesis-cpp, ember, firefox, freeshow, gnome-3-38-2004, gtk-common-themes, lxd, snapcraft, snapd-desktop-integration, snap-store, ubuntu-desktop-installer, subiquity.
- Snaps compliant: TBC
- Snaps not compliant: TBC

Principle 6: sources for published builds are retained and publicly available

This is not currently the case, although the required metadata may be available internally. Implementation will be required in Launchpad/Snap Store.

Principle 7: published on all architectures supported by Ubuntu

- Individual analysis pending: chromium, cyphesis-cpp, ember, firefox, freeshow, gnome-3-38-2004, gtk-common-themes, lxd, snapcraft, snapd-desktop-integration, snap-store, ubuntu-desktop-installer, subiquity.
- Snaps compliant: TBC
- Snaps not compliant: TBC

Principle 8: a public issue/bug tracker is available

- Individual analysis pending: chromium, cyphesis-cpp, ember, firefox, freeshow, gnome-3-38-2004, gtk-common-themes, lxd, snapcraft, snapd-desktop-integration, snap-store, ubuntu-desktop-installer, subiquity.
- Snaps compliant: TBC
- Snaps not compliant: TBC

Unresolved Issues

The following issues are relevant to this document, and probably contradict this policy. In order to make progress, we will accept these for now, and hope to find a suitable resolution later.

Presentation of snaps as equivalent alternatives to debs

William Grant commented:

> *As written this seems to apply to Ubuntu Software and command-not-found, which generally present snaps as equivalent in trust and supportability to debs, for the entire contents of the Snap Store.*
>
> *I agree this is an issue, but I wonder if this wants to be reworded so it just requires that they be clearly presented as second-class citizens, or something like that.*

This identifies a similar issue in two different software components:

1. Ubuntu Software permits users to discover software, which may be a deb from the Ubuntu archive, or may be a snap. These are presented as equal. In the case of a deb, the package would be in compliance with the requirements of this document. In the case of a snap, this compliance is not required. Users may prefer the snap, for example because it appears more up-to-date. But in that case, it may lead to mismatched expectations when compared to Ubuntu's standards as documented here.
2. command-not-found can also be used to discover software, and in this case it may present deb or snap options. When this occurs, exactly the same concern arises as described in the previous point.

Explicit user opt-in for restricted and multiverse

Sebastian Bacher asked:

> *Ubuntu has a restricted section for things like nvidia drivers which are provided by default and non dfsg-free, we should probably have an equivalent exception for snaps?*

Robie Basak responded:

> *AIUI, these always require explicit user opt-in. Eg. the 22.04.1 installer says "Install third-party software for graphics and Wi-Fi hardware and additional media formats. This software is subject to license terms included with its documentation. Some is proprietary."*

> *I think this puts the restricted archive component out of the scope of these requirements, based on the user's explicit informed consent that matches the requirement in the "Out of Scope" section of this document.*
>
> *If there are specific cases where this is needed for snap, can we start by listing them please? Then we can consider adding exceptions for them.*

However, on checking, the restricted and multiverse archive components are enabled on (for example) a default installation of 22.04.1 without that checkbox having being checked. It isn’t clear if this is a mistake or a deliberate decision from the past. If the former, this should be fixed. If the latter, and we conclude that the prior decision should stand, then this should become a documented exception.

Access to build logs and source trees for snaps

Sebastian Bacher asked, in reference to the requirement to make publicly available the sources and build logs, in respect of snaps:

> *We don't have logs compliant with those requirements today afaik, build-snapcraft.io doesn't publish build history and logs and launchpad only lists the 10 most recent builds entries and has no UI to browser older items. Was there any discussion started with the launchpad team about the topic?*

Robie: yes, I did consult with the Snap Store team that this should in principle be fixed at the Snap Store end. The implementation of this requirement in the case of snaps will remain incomplete for now, pending the necessary engineering work. But this document initially defines that our goal as a project is that this will be done.

Other Feedback

There was much valuable feedback received, and this led to many iterations of this document. However some feedback was considered out of scope or deliberately not acted upon. To avoid forgetting about it, these are documented here.

Deferred for possible implementation in the future

William Grant asked, in the context of requiring public sources and build logs to be available: *Do we want to require debug symbols too?*