drm-intel Committer Guidelines

This document describes the detailed merge criteria and committer guidelines for drm-intel. The same criteria and guidelines apply equally to both committers and maintainers.

High Level Guidelines

This section describes the guidelines for pushing patches. Strict rules covering all cases are impossible to write and follow. We put a lot of trust in the sound judgement of the people with commit access, and that only develops with experience. These guidelines are primarily for the committers to aid in making the right decisions, and for others to set their the expectations right.

The short list:

- Only push patches changing *drivers/gpu/drm/i915*.
- Only push patches to *drm-intel-next-queued* branch.
- Ensure certain details are covered, see separate list below.
- You have confidence in the patches you push, proportionate to the complexity and impact they have. The confidence must be explicitly documented with special tags (Reviewed-by, Acked-by, Tested-by, Bugzilla, etc.) in the commit message. This is also your insurance, and you want to have it when the commit comes back haunting you. The complexity and impact are properties of the patch that must be justified in the commit message.
- Last but not least, especially when getting started, you can't go wrong with asking or deferring to the maintainers. If you don't feel comfortable pushing a patch for any reason (technical concerns, unresolved or conflicting feedback, management or peer pressure, or anything really) it's best to defer to the maintainers. This is what the maintainers are there for.
- After pushing, please reply to the list that you've done so.

Detail Check List

An inexhaustive list of details to check:

- The patch conforms to Documentation/process/submitting-patches
- The commit message is sensible, and includes adequate details and references.
- Bug fixes are clearly indicated as such.
- IGT test cases, if applicable, must be complete and reviewed. Please see details on testing requirements.
- The patch series has passed CI pre-merge testing. See CI details below.
- An open source userspace, reviewed and ready for merging by the upstream project, must be available for new kernel ABI. Please see details on upstreaming requirements.
- Relevant documentation must be updated as part of the patch or series.
- Patch series builds and is expected to boot every step of the way, i.e. is bisectable.
- Each patch is of a sensible size. A good rule of thumb metric is, would you (or the author) stand a chance to fairly quickly understand what goes wrong if the commit is reported to cause a regression?
- When pushing someone else's patch you must add your own signed-off per http://developercertificate.org/. dim apply-branch should do this automatically for you.
- For patches that move around lots of code (file rename or extraction) please coordinate with maintainers to avoid unnecessary pain with conflicts. Usually some explicit merges are needed to avoid git getting lost.
- As a general rule, do not modify the patches while applying, apart from the commit message. If the patch conflicts, or needs to be changed due to review, have the author rebase, update and resend. Any change at this stage is a potential issue bypassing CI.

  At most, minor comment and whitespace tweaks are acceptable.
- Please notify maintainers (IRC is fine) of new merge conflicts during drm-tip rebuild, so that they can do backmerges as needed.

On Confidence, Complexity, and Transparency

- Reviewed-by/Acked-by/Tested-by must include the name and email of a real person for transparency. Anyone can give these, and therefore you have to value

them according to the merits of the person. Quality matters, not quantity. Be suspicious of rubber stamps.
- Reviewed-by/Acked-by/Tested-by can be asked for and given informally (on the list, IRC, in person, in a meeting) but must be added to the commit.
- Reviewed-by. All patches must be reviewed, no exceptions. Please see "Reviewer's statement of oversight" in Documentation/process/submitting-patches and review training.
- Acked-by. Indicates acceptance. No reply is not a sign of acceptance, unless you've involved the person (added Cc:, explicitly included in discussion).
- Tested-by. Please solicit separate Tested-by especially from the bug reporters, or people with specific hardware that you or the author do not have.
- There must not be open issues or unresolved or conflicting feedback from anyone. Clear them up first. Defer to maintainers as needed.
- For patches that are simple, isolated, non-functional, not visible to userspace, and/or are in author or reviewer's domain of expertise, one reviewer is enough. Author with commit access can push after review, or reviewer with commit access can push after review.
- The more complicated the patch gets, the greater the impact, involves ABI, touches several areas or platforms, is outside of author's domain of expertise, the more eyeballs are needed. For example, several reviewers, or separate author, reviewer and committer. Make sure you have experienced reviewers. Involve the domain expert(s). Possibly involve people across team/group boundaries. Possibly involve the maintainers. Give people more time to voice their concerns. Aim for what's described below in more detail as "rough consensus".
- Most patches fall somewhere in between. You have to be the judge, and ensure you have involved enough people to feel comfortable if the justification for the commit is questioned afterwards.

On Rough Consensus

For really big features with a lot of impact on the code, new cross-driver ABI (like new atomic properties), or features that will take a few patch series (and maybe a few kernel releases) aim for rough consensus among a wide group of stakeholders. There's three components for that:

- Identify stakeholders beyond just the patch author and reviewers. This includes contributors with experience in code ancillary to your feature, domain experts, maybe maintainers. Also include maintainers and reviewers of the userspace

component for new ABI, which often means non-Intel people. In case of doubt ask maintainers for a reasonable list of people. Make sure you gather their input actively, don't expect them to deliver it on their own - most are really busy.
- Have agreement among all these stakeholders what the code should look like in the end. Generally disagreement on the end state is considered a blocker, and overruling such disagreement by maintainers is done only in exceptional cases. Generally what happens is that maintainers instead do all the work of soliciting input, which doesn't scale and should be the patch author's and reviewer's duty. Try to document this consensus somewhere, either in a summary email, interface patch to describe the data structures and vfuncs. Or maybe even a more formal design spec, although in the past that didn't work out when non-Intel people are involved, and the actually merged interface differed from the spec a lot. Just assumed consensus on IRC tends to result in misunderstandings.
- Have a concrete plan how to get to the agreed end state in the codebase. Often it makes sense to have an intermediate patch set merged first, and then polish it in tree. Or merge new ABI in stages. Usually this means that a new feature or ABI is disabled by default at first. For the actual plan some unhappiness from stakeholders about the execution is acceptable, as long as the overall stability and other ongoing work isn't negatively affected. As a rule of thumb new ABI or features should never be in a partial/limbo stage for more than 2-3 kernel releases. If that seems unlikely, more work should happen pre-merge.

Try to reach rough consensus before spending months writing code you might need to throw away or at least entirely rewrite again. Also make sure that all discussions happen in public forums, and make sure there's a searchable permanent record of any discussions for later reference. This means that for most things internal meetings are not the most suitable venue.

Continuous Integration and Pre-Merge Testing

The requirements for CI pre-merge testing are:

- `checkpatch.pl` does not complain. (Some of the more subjective warnings may be ignored at the committer's discretion.)
- The patch does not introduce new `sparse` warnings.
- Patch series must pass IGT Basic Acceptance Tests (BAT) on all the CI machines without causing regressions.

- Patch series must pass full IGT tests on CI shard machines without causing regressions.
- Patch series must pass GPU piglit tests on all CI machines without causing regressions.

The CI bots will send results to the patch author and intel-gfx for any patches tracked by patchwork. The results are also available on patchwork and the CI site.

Check CI failures and make sure any sporadic failures are a) pre-existing, and b) tracked in bugzilla. If there's anything dubious that you can't track down to pre-existing and tracked issues please don't push, but instead figure out what's going on.

Labeling Fixes Before Pushing

To label fixes that should be cherry-picked to the current -rc development kernel or drm-next, the commit message should have the 'Fixes:' tag.

If the fix is relevant for a released kernel please also

> Cc: stable@vger.kernel.org

'Fixes:' tag needs to added to any patch fixing a regression or an incorrect behavior from previous patches. This tag help maintainers and tools to decide where to cherry-pick a patch to. This also extremely useful for product groups to know which bugfixes they must include. To follow the recommended format please generate the Fixes: line using

```
$ dim fixes $regressing_commit
```

This will also add the correct Cc: line if one is needed.

If the Cc: or Fixes: was forgotten, you can still reply to the list with that, just like any other tags, and they should be picked up by whoever pushes the patch.

The maintainers will cherry-pick labeled patches from drm-intel-next-queued to the appropriate branches.

‘Fixes:’ tag is described in Documentation/process/submitting-patches.

Requesting Fixes Cherry-Pick Afterwards

It’s not uncommon for a patch to have been committed before it’s identified as a fix needing to be backported.

If the patch is already in Linus’ tree, please follow stable kernel rules.

Otherwise, send an email to intel-gfx@lists.freedesktop.org containing the subject of the patch, the commit id, why you think it should be applied, and what branch you wish it to be applied to.

Replying to the original patch is also fine, but please mention the commit id.

Alternatively, if the cherry-pick has conflicts, please send a patch to intel-gfx@lists.freedesktop.org with subject prefix “drm-intel-fixes PATCH” or “drm-intel-next-fixes PATCH” depending on the branch. Please add ‘git cherry-pick -x’ style annotation above your Signed-off-by: line in the commit message:

> (cherry picked from commit 0bff4858653312a10c83709e0009c3adb87e6f1e)

Tooling

drm-intel git repositories are managed with dim.