Zweibach - Text

The various Hydrus guides I've written on my github repo, copied here for consolidation.

- PTR

PTR

 # PTR for Dummies
or
Myths and facts about the Public Tag Repository

What is the PTR?
Short for **P**ublic **T**ag **R**
epository, a now community managed repository of tags. Locally it acts as a tag service, just like `my tags`
. At the time of writing 54 million files have tags on it. The PTR only store the sha256 hash and tag mappings of a file, not the files themselves or any non-
tag meta data. In other words: If you don not see it in the tag list then it is not stored.

Most of the things in this document also applies to [self-hosted servers](https://hydrusnetwork.github.io/hydrus/help/server.html), except for tag guidelines.

Connecting to the PTR
The easiest method is to use the built in function, found under `help -> add the public tag repository`
. For adding it manually, if you so desire, read the Hydrus help document on [access keys](https://hydrusnetwork.github.io/hydrus/help/access_keys.html).
If you are starting out completely fresh you can also download a fully synced client [here](https://koto.reisen/quicksync/
). Though possibly a bit (couple of days or so usually) out of date it will none the less save time. Some settings may differ from the defaults of an official installation.
Once you are connected Hydrus will proceed to download and then process the update files. The progress of this can be seen under `services -> review services -> remote -> tag repositories -> public tag repository`
. Here you can view its status, your account (the default account is a shared public account. Currently only janitors and the administrator have personal accounts), tag status, and how synced you are. Being behind on the sync by a certain amount makes you unable to push tags and petitions until you are caught up again.

How does it work?

For something to end up on the PTR it has to be pushed there. Tags can either be entered into the tag service manually by the user through the `manage tags` window, or be routed there by a parser when downloading files. See [parsing tags](https://hydrusnetwork.github.io/hydrus/help/getting_started_downloading.html). Once tags have been entered into the PTR tag service they are pending until pushed. This is indicated by the `pending ()` that will appear between `tags` and `help` in the menu bar. Here you can chose to either push your changes to the PTR or discard them.

- Adding tags pass automatically.
- Deleting (petitioning) tags requires janitor action.
- If a tag has been deleted from a file it will not be added again.
- Currently there is no way for a normal user to re-add a deleted tag. If it gets deleted then it is gone. A janitor can undelete tags manually.
- Adding and petitioning siblings and parents all require janitor action.
- The client always assumes the server approves any petition. If your petition gets rejected you won't know.

When making petitions it is important to remember that janitors are only human. We do not necessarily know everything about every niche. We do not necessarily have the files you are making changes for and we will only see a blank thumbnail if we do not have the file. Explain why you are making a petition. Try and keep the number of files manageable. If a janitor at any point is unsure if the petition is correct they are likely to deny the entire petition rather than risk losing good tags. Some users have pushed changes regarding hundreds of tags over thousands of files at once, but due to disregarding PTR tagging practices or being lazy with justification the petition has been denied entirely. Or they have just been plain wrong, trying to impose frankly stupid tagging methods.

Furthermore, if you are two weeks out of sync with PTR you are unable to push additions or deletions until you're back within the threshold.

`Q. Does this automagically tag my files?`
`A. No. Until we get machine learning based auto-tagging nothing is truly automatic. All tags on the PTR were uploaded by another user, so if nobody uploaded tags associated with the hash of your file it won't have any tags in the PTR.`

`Q. How good is the PTR at tagging [insert file format or thing from site here]?`
`A. That depends largely on if there's a scrapable database of tags for whatever you're asking about. Anything that comes from a booru or site that supports tags is fairly likely to have something on the PTR. Original content on some obscure chan-style imageboard is less so.`

`Q. Help! My files don't have any tags! What do!?`
`A. As stated above, some things are just very likely to not have any tags. It is also possible that the files have been altered by whichever service you downloaded from. Imgur, Reddit, Discord, and many other sites and services recompress images to save space which might give it a different hash even if it looks indistinguishable from the original file. Use one of the IQDB lookup programs linke

d in` [Cuddle's wiki](https://github.com/CuddleBear92/Hydrus-Presets-and-Scripts/wiki/0-Hydrus-Apps-and-Scripts).

`Q. Why is my database so big!? This can't be right.`
`A. It is working as intended. The size is because you are literally downloading and processing the entire tag database and history of the PTR. It is done this way to ensure redundancy and privacy. Redundancy because anybody with an up-to-
date PTR sync can just start their own. Privacy because nobody can tell what files you have since you are downloading the tags for everything the PTR has.`

`Q. Does that mean I can't do anything about the size?`
`A. Correct. There are some plans to crunch the size through a few methods but there are a lot of other far more requested features being, well, requested. Speaking crassly if you are bothered by the size requirement of the PTR you probably don't have a big enough library to really benefit and would be better of just using the IQDB script.`

Janitors
Janitors are the people that review petitions. You can meet us at the community [Discord](https://discord.gg/3H8UTpb
) to ask questions or see us bitch about some of the silly stuff boorus and users cause to end up in the PTR.

Tag Guidelines

These are a mix of standard practice used by various boorus and changes made by Hydrus Developer and PTR users, ratified by the janitors that actually have to manage all of this. The "full" document is viewable at [Cuddle's git repo](https://raw.githubusercontent.com/CuddleBear92/Hydrus-Presets-and-Scripts/master/tag%20guidelines). See Hydrus Developer's [
thoughts on a public tagging schema](
https://hydrusnetwork.github.io/hydrus/help/tagging_schema.html).

If you are looking to help out by tagging low tag-
count files, remember to keep the tags objective, start simple by for example adding the characters/persons and big obvious things in the image or what else. Tagging every little thing and detail is a sure path to burnout.
If you are looking to petition removal of tags then it is preferable to sibling common misspellings, underscores, and defunct tags rather than deleting them outright. The exception is for ambiguous tags where it is better to delete and replace with a less ambiguous tag. When deleting tags that don't belong in the image it can be helpful if you include a short description as to why.
It's also helpful if you sanitise downloaded tags from sites with tagged galleries before pushing them to the PTR. For example Pixiv, where you can have a gallery of multiple images, each containing one character, and all of the characters being tagged. Consequently all images in that gallery will have all of the character tags despite no image having more than one character.

Siblings and parents

When making siblings, go for the closest less-bad tag. Example: `bad_tag` -> `bad tag`
, rather than going for what the top level sibling might be. This creates less potential future work in
 case standards change and makes it so your request is less likely to be denied by a janitor not bei
ng entirely certain that what you're asking is right. Be careful about creating siblings for potentially
 ambiguous tags. Is `james bond` supposed to be `character:james bond` or is it
`series:james bond`
? This is a bit of a bad example due to having the case of the character always belonging to the ser
ies, so you can safely sibling it to `series:james bond`
 since all instances of the character will also have the series, but not all instances of the series will
have the character. So let us look at another example: how about `wool`
? Is it the material harvested from sheep, or is it the Malaysian artist that likes to draw Touhou? In
doubtful cases it's better to leave it as is, petition the tag for deletion if it's incorrect and add the co
rrect tag.

When making parents, make sure it's an always factually correct relationship.
`character:james bond` always belongs to `series:james bond`. But `character:james bond`
 is not always `person:pierce brosnan`. Common examples of not-
always true relationships: gender (genderbending), species (furrynisation/humanisation/anthropom
orphism), hair colour, eye colour, and other mutable traits.

Namespaces

`creator:`
 Used for the creator of the tagged piece of media. Hydrus being primarily used for images it will of
ten be the artist that drew the image. Other potential examples are the author of a book or musicia
n for a song.
`character:` Refers to characters. James Bond is a character.
`person:` Refers to real persons. Pierce Brosnan is a person.
`series:` Used for series. **James Bond** is a series tag and so is **GoldenEye**
. Due to usage being different on some boorus chance is that you will also see things like Absolut V
odka and other brands in it.
`photoset:`
 Used for photosets. Primarily seen for content from idols, cosplayers, and gravure idols.
`studio:`
 Is used for the entity that facilitated the production of the file or what's in it. Eon Productions for th
e **James Bond** movies.
`species:`
 Species of the depicted characters/people/animals. Somewhat controversial for being needlessly d
etailed, some janitors not liking the namespace at all. Primarily used for furry content.
`title:`
 The title of the file. One of the tags Hydrus uses for various purposes such as sorting and collectin
g. Somewhat tainted by rampant Reddit parsers.
`medium:`
 Used for tags about the image and how it's made. Photography, water painting, napkin sketch as a
 few examples. White background, simple background, checkered background as a few others.
****What you see about the image.****
`meta:`
 This namespace is used for information that isn't visible in the image itself or where you might nee

d to go to the source. Some examples include: third-
party edit, paid reward (patreon/enty/gumroad/fantia/fanbox), translated, commentary, and such.
What you know about the image.

Namespaces not listed above are not "supported" by the janitors and are liable to get siblinged out, removed, and/or mocked if judged being bad and annoying enough to justify the work. Do not take this to mean that all un-
listed namespaces are bad, some are created and used by parsers to indicate where an image came from which can be helpful if somebody else wants to fetch the original or check source tags against the PTR tags. But do exercise some care in what you put on the PTR if you use custom namespaces. Recently `clothing:` was removed due to being disliked, no booru using it, and the person(s) pushing for it seeming to have disappeared, leaving a less-than-
finished mess behind. It was also rife with lossy siblings and things that just plain don't belong with clothing, such as `clothing:brown hair`.