Misc

- Privacy
- Contact and Links
- Financial Support
- FAQ

Privacy

privacy

Repositories are designed to respect your privacy. They never know what you are searching for. The client synchronises (copies) the repository's entire file or mapping list to its internal database, and does its own searches over those internal caches, all on your hard drive. It *never* sends search queries outside your own computer, nor does it log what you do look for. Your searches are your business, and no-one else's.

The PTR has a public shared access key. You do not have to contact anyone to get the key, so no one can infer who you are from it, and all regular user uploads are merged together, making it all a big mess. The PTR is more private than this document's worst case scenarios.

The only privacy risk for hydrus's repositories are in what you upload (ultimately by using the pending menu at the top of the program). Even then, it would typically be very difficult even for an admin to figure anything about you, but it is possible.

Repositories know nothing more about your client than they can infer from what you choose upload, and the software usually commands them to forget as much as possible as soon as possible. Specifically:

	tag repository		**file repository**	
	upload mappings	**download mappings**	**upload file**	**download file**
Anonymous account is linked to action	Yes	No	Yes	No
IP address is remembered	No	No	Maybe	No

i.e:

- If you download anything from any repository, your accessing it will not be recorded. A running total of your approximate bandwidth and number of queries made for the current month *is* kept so the respective administrator can combat leechers.
- If you upload a mapping to a tag repository, your anonymous account is linked so the administrator can quickly revoke all of a rule-breaker's contributions. Your IP address is

forgotten.
- If you upload a file to a file repository, your anonymous account is linked so the administrator can quickly revoke all of a rule-breaker's contributions. Your IP may be recorded, depending on whether the repository's administrator has decided to enable ip upload-logging or not.

Furthermore:

- Administrators for a particular repository can see which accounts uploaded what. If IP addresses are available, they can discover which IP uploaded a particular file, and when.
- Repositories do not talk to each other.
- All accounts are anonymous. Repositories do not *know* any of their accounts' access keys and cannot produce them on demand; they can determine whether a particular access key refers to a particular account, but the access keys themselves are all irreversibly hashed inside the repository database.

As always, there are some clever exceptions, mostly in servers between friends that will just have a handful of users, where the admin would be handing out registration keys and, with effort, could pick through the limited user creation records to figure out which access key you were. In that case, if you were to tag a file three years before it surfaced on the internet, and the admin knew you are attached to the account that made that tag, they could infer you most likely created it. If you set up a file repository for just a friend and yourself, it becomes trivial by elimination to guess who uploaded the NarutoXSonichu shota diaper fanon. If you sign up for a file repository that hosts only certain stuff and rack up a huge bandwidth record for the current month, anyone who knows that and also knows the account is yours alone will know basically what you were up to.

The PTR has a shared access key that is already public, so the risks are far smaller. No one can figure out who you are from the access key.

Note that the code is freely available and entirely mutable. If someone wants to put the time in, they could create a file repository that looks from the outside like any other but nonetheless logs the IP and nature of every request. As with any website, protect yourself, and if you do not trust an admin, do not give them or their server any information about you.

Even anonymised records can reveal personally identifying information. Don't trust anyone on any site who plans to release internal maps of 'anonymised' accounts -> content, even for some benevolent academic purpose.

Contact and Links

contact and links

I welcome all your bug reports, questions, ideas, and comments. It is always interesting to see how other people are using my software and what they generally think of it. Most of the changes every week are suggested by users.

You can contact me by email, twitter, tumblr, discord, or the 8chan.moe /t/ thread or Endchan board--I do not mind which. Please know that I have difficulty with social media, and while I try to reply to all messages, it sometimes takes me a while to catch up.

The Github Issue Tracker was turned off for some time, as it did not fit my workflow and I could not keep up, but it is now running again, managed by a team of volunteer users. Please feel free to submit feature requests there if you are comfortable with Github. I am not socially active on Github, and it is mostly just a mirror of my home dev environment, where I work alone.

I am on the discord on Saturday afternoon, USA time, if you would like to talk live, and briefly on Wednesday after I put the release out. If that is not a good time for you, feel free to leave me a DM and I will get to you when I can. There are also plenty of other hydrus users who idle who would be happy to help with any sort of support question.

I delete all tweets and resolved email conversations after three months. So, if you think you are waiting for a reply, or I said I was going to work on something you care about and seem to have forgotten, please do nudge me.

Anyway:

- homepage
- github
- issue tracker
- 8chan.moe /t/ (Hydrus Network General) (endchan bunker (.org))
- tumblr (rss)
- new downloads
- old downloads
- twitter
- email

- discord
- patreon
- user-run wiki (including download presets for several non-default boorus)

Financial Support

can I contribute to hydrus development?

I do not expect anything from anyone. I'm amazed and grateful that anyone wants to use my software and share tags with others. I enjoy the feedback and work, and I hope to keep putting completely free weekly releases out as long as there is more to do.

That said, as I have developed the software, several users have kindly offered to contribute money, either as thanks for a specific feature or just in general. I kept putting the thought off, but I eventually got over my hesitance and set something up.

I find the tactics of most internet fundraising very distasteful, especially when they promise something they then fail to deliver. I much prefer the 'if you like me and would like to contribute, then please do, meanwhile I'll keep doing what I do' model. I support several 'put out regular free content' creators on Patreon in this way, and I get a lot out of it, even though I have no direct reward beyond the knowledge that I helped some people do something neat.

If you feel the same way about my work, I've set up a simple Patreon page here. If you can help out, it is deeply appreciated.

FAQ

what is a repository?

A *repository* is a service in the hydrus network that stores a certain kind of information--files or tag mappings, for instance--as submitted by users all over the internet. Those users periodically synchronise with the repository so they know everything that it stores. Sometimes, like with tags, this means creating a complete local copy of everything on the repository. Hydrus network clients never send queries to repositories; they perform queries over their local cache of the repository's data, keeping everything confined to the same computer.

what is a tag?

wiki

A *tag* is a small bit of text describing a single property of something. They make searching easy. Good examples are "flower" or "nicolas cage" or "the sopranos" or "2003". By combining several tags together (e.g. ['tiger woods', 'sports illustrated', '2008'] or ['cosplay', 'the legend of zelda']), a huge image collection is reduced to a tiny and easy-to-digest sample.

A good word for the connection of a particular tag to a particular file is *mapping*.

Hydrus is designed with the intention that tags are for *searching*, not *describing*. Workflows and UI are tuned for finding files and other similar files (e.g. by the same artist), and while it is possible to have nice metadata overlays around files, this is not considered their chief purpose. Trying to have 'perfect' descriptions for files is often a rabbit-hole that can consume hours of work with relatively little demonstrable benefit.

All tags are automatically converted to lower case. 'Sunset Drive' becomes 'sunset drive'. Why?

1. Although it is more beautiful to have 'The Lord of the Rings' rather than 'the lord of the rings', there are many, many special cases where style guides differ on which words to capitalise.
2. As 'The Lord of the Rings' and 'the lord of the rings' are semantically identical, it is natural to search in a case insensitive way. When case does not matter, what point is there in recording it?

Furthermore, leading and trailing whitespace is removed, and multiple whitespace is collapsed to a single character.

```
'  yellow   dress '
```

becomes

```
'yellow dress'
```

what is a namespace?

A *namespace* is a category that in hydrus prefixes a tag. An example is 'person' in the tag 'person:ron paul'--it lets people and software know that 'ron paul' is a name. You can create any namespace you like; just type one or more words and then a colon, and then the next string of text will have that namespace.

The hydrus client gives namespaces different colours so you can pick out important tags more easily in a large list, and you can also search by a particular namespace, even creating complicated predicates like 'give all files that do not have any character tags', for instance.

why not use filenames and folders?

As a retrieval method, filenames and folders are less and less useful as the number of files increases. Why?

- A filename is not unique; did you mean this "04.jpg" or *this* "04.jpg" in another folder? Perhaps "04 (3).jpg"?
- A filename is not guaranteed to describe the file correctly, e.g. hello.jpg
- A filename is not guaranteed to stay the same, meaning other programs cannot rely on the filename address being valid or even returning the same data every time.
- A filename is often--for *ridiculous* reasons--limited to a certain prohibitive character set. Even when utf-8 is supported, some arbitrary ascii characters are usually not, and different localisations, operating systems and formatting conventions only make it worse.

- Folders can offer context, but they are clunky and time-consuming to change. If you put each chapter of a comic in a different folder, for instance, reading several volumes in one sitting can be a pain. Nesting many folders adds navigation-latency and tends to induce less informative "04.jpg"-type filenames.

So, the client tracks files by their *hash*. This technical identifier easily eliminates duplicates and permits the database to robustly attach other metadata like tags and ratings and known urls and notes and everything else, even across multiple clients and even if a file is deleted and later imported.

As a general rule, I suggest you not set up hydrus to parse and display all your imported files' filenames as tags. 'image.jpg' is useless as a tag. Shed the concept of filenames as you would chains.

can the client manage files from their original locations?

When the client imports a file, it makes a quickly accessible but human-ugly copy in its internal database, by default under *install_dir/db/client_files*. When it needs to access that file again, it always knows where it is, and it can be confident it is what it expects it to be. It never accesses the original again.

This storage method is not always convenient, particularly for those who are hesitant about converting to using hydrus completely and also do not want to maintain two large copies of their collections. The question comes up--"can hydrus track files from their original locations, without having to copy them into the db?"

The technical answer is, "This support could be added," but I have decided not to, mainly because:

- Files stored in locations outside of hydrus's responsibility can change or go missing (particularly if a whole parent folder is moved!), which erodes the assumptions it makes about file access, meaning additional checks would have to be added before important operations, often with no simple recovery.
- External duplicates would not be merged, and the file system would have to be extended to handle pointless 1->n hash->path relationships.
- Many regular operations--like figuring out whether orphaned files should be physically deleted--are less simple.
- Backing up or restoring a distributed external file system is much more complicated.
- It would require more code to maintain and would mean a laggier db and interface.
- Hydrus is an attempt to get *away* from files and folders--if a collection is too large and complicated to manage using explorer, what's the point in supporting that old system?

It is not unusual for new users who ask for this feature to find their feelings change after getting more experience with the software. If desired, path text can be preserved as tags using regexes during import, and getting into the swing of searching by metadata rather than navigating folders often shows how very effective the former is over the latter. Most users eventually import most or all of their collection into hydrus permanently, deleting their old folder structure as they go.

For this reason, if you are hesitant about doing things the hydrus way, I advise you try running it on a smaller subset of your collection, say 5,000 files, leaving the original copies completely intact. After a month or two, think about how often you used hydrus to look at the files versus navigating through folders. If you barely used the folders, you probably do not need them any more, but if you used them a lot, then hydrus might not be for you, or it might only be for some sorts of files in your collection.

why use sqlite?

Hydrus uses SQLite for its database engine. Some users who have experience with other engines such as MySQL or PostgreSQL sometimes suggest them as alternatives. SQLite serves hydrus's needs well, and at the moment, there are no plans to change.

Since this question has come up frequently, a user has written an excellent document talking about the reasons to stick with SQLite. If you are interested in this subject, please check it out here:

https://gitgud.io/prkc/hydrus-why-sqlite/blob/master/README.md

what is a hash?

wiki

Hashes are a subject you usually have to be a software engineer to find interesting. The simple answer is that they are unique names for things. Hashes make excellent identifiers inside software, as you can safely assume that f099b5823f4e36a4bd6562812582f60e49e818cf445902b504b5533c6a5dad94 refers to one particular file and no other. In the client's normal operation, you will never encounter a file's hash. If you want to see a thumbnail bigger, double-click it; the software handles the mathematics.

For those who are *interested: hydrus uses SHA-256, which spits out 32-byte (256-bit) hashes. The software stores the hash densely, as 32 bytes, only encoding it to 64 hex characters when the user views it or copies to clipboard. SHA-256 is not perfect, but it is a great compromise candidate; it is secure for now, it is reasonably fast, it is available for most programming languages, and newer CPUs perform it more efficiently all the time.*

what is an access key?

The hydrus network's repositories do not use username/password, but instead a single strong identifier-password like this:

7ce4dbf18f7af8b420ee942bae42030aab344e91dc0e839260fcd71a4c9879e3

These hex numbers give you access to a particular account on a particular repository, and are often combined like so:

7ce4dbf18f7af8b420ee942bae42030aab344e91dc0e839260fcd71a4c9879e3@hostname.com:45871

They are long enough to be impossible to guess, and also randomly generated, so they reveal nothing personally identifying about you. Many people can use the same access key (and hence the same account) on a repository without consequence, although they will have to share any bandwidth limits, and if one person screws around and gets the account banned, everyone will lose access.

The access key is the account. Do not give it to anyone you do not want to have access to the account. An administrator will never need it; instead they will want your *account key*.

what is an account key?

This is another long string of random hexadecimal that *identifies* your account without giving away access. If you need to identify yourself to a repository administrator (say, to get your account's permissions modified), you will need to tell them your account key. You can copy it to your clipboard in *services->review services*.

why can my friend not see what I just uploaded?

The repositories do not work like conventional search engines; it takes a short but predictable while for changes to propagate to other users.

The client's searches only ever happen over its local cache of what is on the repository. Any changes you make will be delayed for others until their next update occurs. At the moment, the update period is 100,000 seconds, which is about 1 day and 4 hours.