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.

Revision #1
Created 12 March 2021 16:54:48 by CuddleBear
Updated 12 March 2021 18:24:07 by CuddleBear