

Tana Paste

[Tana](#) • 6 mins

[May 10th 2024](#)

Generate rich Tana objects through a simple plain text language based on markdown.

Tana Paste is a way of creating rich Tana structures like fields, tags, dates, checkboxes etc, through a simple plain text paste.

It is not designed for inputting large amounts of information, and rather for automating one-off, small, generative results. For heavy lifts, use import tools, API or other scalable solutions.

It is also not designed to update or change existing information on nodes and fields. While you can specify existing template supertags and fields to use, it can only only generate new information within them.

The syntax is loosely modelled on Markdown.

A Tana Paste is a plain text paste that begins with the string `%%tana%%`.

Nodes are prepended by `-` (a dash and a space) and indentation indicates children.

References are defined with double brackets `[[]]`


- This will look for a node in the Library of the current workspace matching that name. If none exists, a new node will be created and

linked to. Multiple links with the same string will always point at the same node.

- You can also specify a node target using `[[link name^nodeID]]`. If the `nodeID` does not exist, it will fall back to using the link name.
- If you only supply a `nodeID`, and it does not exist, it will create a link to "undefined", which you can later rename.
- If you also specify the tag of the link, like so: `[[Wes Anderson #person]]`, it will find a node named Wes Anderson and tagged #person anywhere in your graph, and link to that, or create a new node in the library with the provided tag.

We do not support more than one inline reference to a given node, from a given node.

Fields use `::` between the field name and the value.

- Fields will look up by string, but if they are nested underneath a supertag which has any fields with that name, Tana will prioritize the field from the supertag definition.
-  stricter requirements for matching existing fields:
 - `^nodeID::field value`
This will work if `nodeID` is a field that exists in a loaded workspace
 - `someTitle^nodeID::field value`
If `nodeID` is a field in a workspace that is not loaded this will not work, use `[[someTitle^nodeID]]::field value` instead
 - `foo^bar::field value`
This will not work, since `bar` is not an actual `nodeID` that

exists

- If the field has a type of option or instance, and the provided value matches perfectly with an existing option or instance, the existing option or instance will be linked even if the text in the Tana Paste is not a link.
- Fields with multiple values can be specified through `- [[field name]]::` and then the values nested and indented underneath (also prepended with dashes)

Tags use `#`, and multi-word tag names can optionally be wrapped in `[[]]`, so either `#bug` or `#[[my ideas]]`.

- If no tag exists, a new one will be created.
- If you want to specify a specific tag, use `^` like `#bug^nodeID`. It will fall back to the string name, and if no string is provided and the `nodeID` doesn't match, the tag will not be created.
- You can link to a tag using `[[#tag]]` (especially useful in search nodes)

You can set the view of a node, using `%%view:table%%`, `%%view:cards%%`, `%%view:tabs%%`, or `%%view:calendar%%`. See search node example below for a demo.

Add `%%search%%` to a node to make it a search node, with the search expression indented under.

In a search expression, we recognize system nodes like `Set/Not set` and field operators like `LINKS TO`, `OWNER` etc.

For field names, it's looking for a name match. To minimize confusion, add or replace the name with `^nodeID` to specify the exact field you mean.

Example:

```
%%tana%%
- %%search%% Todo with start-date within next month
%%view:table%%
- [[#task]]
- LT::

- NOT DONE
```

Simple dates look like this: `[[date:2021-02]]`.

- They can include time `[[date:2021-02-01 20:30]]`, or only include the week `[[date:2021-W02]]`, month `[[date:2021-02]]` or year `[[date:2021]]`
- Durations are written like `[[date:2021-02/2021-04-05]]`.
- We also still support the old format `[[August 22nd, 2020]]`.

Checkboxes use `[]` and `[x]` at the beginning of a node to indicate checked or unchecked.


- For fields/nodes which already have a checkbox, providing the `[]` will not change anything, but for other nodes/fields it will insert a checkbox.
- In all cases, `[x]` will correctly set the checkbox as checked.

To add a URL, use this syntax: `[name](url)`. Example:

```
%%tana%%
```

```
- [Tana website](https://tana.inc)
```

The link needs to include the `https://` prefix.

- There is support for Markdown tables and code blocks.
- Images use the syntax ``
- Formatting: `**bold**`, `^^highlighted^^`, `__italic__`
-  **NEW** Headings: Start node with `!!`
- You can find examples of scripts that output text in the Tana Paste format here <https://github.com/tanainc/tana-paste-examples>
- You can now use `sys:nodeId`, `sys:nodeURL` or `sys:tags` on an ancestor node (using `sys:owner` one or multiple times). Use with Insert Tana Paste command to reference ancestor nodes, for example: `[[^${sys:owner.sys:nodeId}]]`, or to include information in AI prompts. ([2025 / wk 33](#))

Examples

- ▼

Yes you can: We have a basic syntax for this which you can [read about here](#).

Example of a task search:

```
%%tana%%
- %%search%% Fei's tasks
  - OR::
    - [[#task]]
    - [[#CS todo]]
  - Assigned to::
    - [[Fei-Ling Tseng]]
  - NOT DONE
```

When I paste the above, this is the result:

- ▼

How to use Tana Paste with supertags that have non-alphanumeric characters?

Tana Paste can usually parse supertags that are one word, such as `#task`. If a supertag has spaces, numbers or irregular characters, try wrapping the name in double brackets like this:

```
#[[1'3$4'6]]
```

If you want to grab the supertag definition, include the hashtag within the brackets, like this:

```
[[#1'3$4'6]]
```

- ▼

How can I add supertags on things that I send from Tana Capture?

While it isn't possible to tag things when you're sending things from Tana Capture, you can have Tana do some post-processing magic to convert written-out tags to real tags once they arrive the Inbox—no AI needed!

1. This method uses a simple [Tana Paste](#) command that you can run on the Inbox node, which targets all new children that have "#" in them.
2. Once you've created the command, you must add it to the **On child added** section of the Inbox node. You can find it by running the command **Debug node** on the title of the Inbox.

"All it does is paste the nodes that have # in them and that are not tagged, using Tana Paste.

Say we add this through Tana Capture:

buy milk #todo

The command will paste that exact text back in with Tana Paste, where **\${name}** is **buy milk #todo** and **\${sys:content}** are any child notes that are present. Using Tana Paste, **#todo** will be added as a tag.

Note: this creates a new node, so the created date will change, meaning you'll lose the time that the node was captured through Tana Capture."

Questions

• ▼

Can a command move the current node to a field of a new node?

Yes, you can. Make a new [command node](#), and use the command `Insert Tana Paste` with the following prompt:

```
- ${name} #note
- Based on:: [[^${sys:nodeId}]]
```

The first line is the new node. It uses the name of the current node to inform the name of the new node by using the title expression `${name}`. The supertag `#note` can be enough for Tana to find the right tag to use, but you can specify it more precisely by adding the `^nodeID` to it. See the [Tana Paste](#) doc for more info on this convention. To retrieve the nodeID of a supertag, run the command `Copy link (HTML formatted)` on a supertag definition and paste the results. It will paste with the nodeID written out:

The second line is the field. After the name of the field and the double-colon `::` is a reference to the current node that the command is being run on, which it is able to target by retrieving the nodeID of the current node using the title expression `${sys:nodeId}`.

This is what the command node should look like:

Special thanks to Navigator Emmanuel Galanos for this clever solution!