Stamps v1.1

By Adrian Pueyo and Alexey Kuchinski

USER GUIDE

Thank you for downloading ***Stamps***. We hope it will help speed up your node workflow while keeping your script more readable and organized.

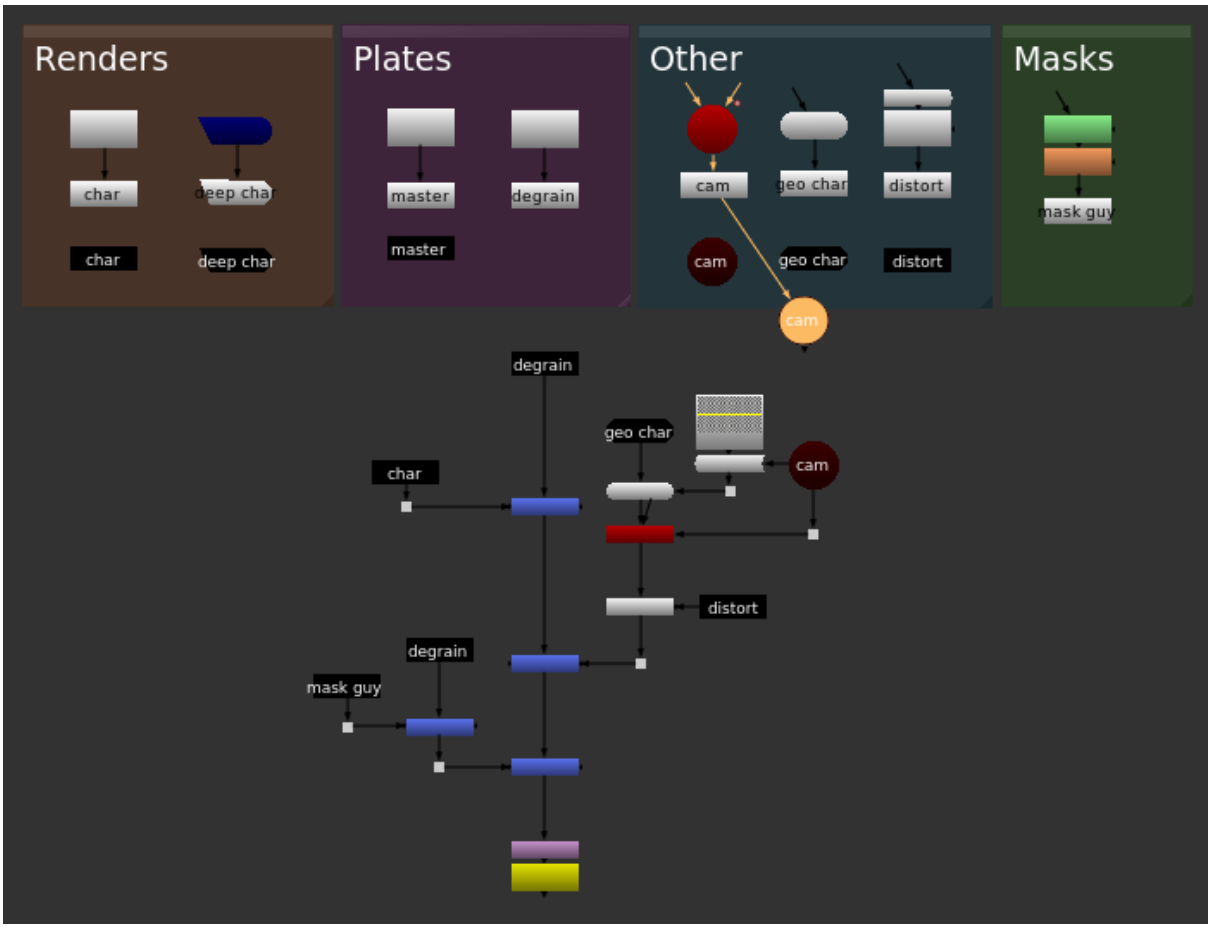
Stamps can be downloaded here: https://github.com/adrianpueyo/stamps/

Complete video tutorial series about Stamps:
http://adrianpueyo.com/stamps

Index

1. Introduction

> ***Stamps is a free and production-ready node connection system for Nuke**, that enables placing the main assets in a single place on the Node Graph, through smart and distinct nodes with hidden inputs that reconnect themselves when needed.*



Stamps encourages a way of working where all the main assets for the shot are placed in a single part of the Node Graph, and you avoid duplicating any Cameras, Rotos, Read nodes, etc. or creating complex arrow connections in order to be able to pipe them into different parts of your script.

- ***Artist:** Compositing speed and efficacy.*
- ***Script:** Improved size and processing speed.*
- ***Pipeline:** Robust and predictable workflow.*

It also makes opening scripts from other people a lot easier as you know you don't need to fish for assets all over the node graph. Most of the major studios have their own approaches at solving this problem through hidden dots, nodes that connect to a camera, or modified PostageStamp nodes, but all the hidden-input systems create new problems that we are proposing a way of solving with ***Stamps*.**

2. Why to use Stamps

We believe the **main benefits** of incorporating ***Stamps*** into a pipeline are:

1. All your main comp assets can be at the top of the script, or in defined places, and you **only load the assets a single time** (reads, cameras, geos, rotos, ...), saving processing time.
2. Sourcing several times from a same input, you can **easily swap versions** when there is a new render or element update.
3. **Auto-Reconnecting** means you don't have to think about copying and pasting nodes (or even node trees) and maintaining their connections, even between different nuke scripts. You can also feel confident that you can always find where the stamp needs to be connected to thanks to the *Advanced Reconnect* functionality.
4. With the **workflow** that the usage of ***Stamps*** encourages, **making changes** on a script from another compositor becomes a lot easier, because:
    a. As **the main assets** (renders, rotos, etc) **are together,** you don't have to go through the entire script checking if you missed something.
    b. The **script** itself is a lot **more readable**, because the amount of non-relevant lines on the script is substantially lower. It avoids dot spiderwebs, and allows for cleaner and smaller comp scripts.
    c. ***Stamps*** **keep their internal id** when belonging to a template, so if a compositor does modifications to a template on a shot and the shot gets approved, those modifications can be transferred painlessly to similar shots through copy-pasting without thinking about reconnecting anything.
5. ***Stamps*** are clearly defined on the node graph as places with hidden inputs, so you also save a lot of time you'd spend creating organized lines and dots.
6. The **tag system** allows for organizing the stamps into categories so you will have super **quick access to every Stamp** from any part of the node graph.
7. Having both Stamps and Anchors makes extremely **clear where the hidden lines are** located, so you never worry about deleting nodes that might have something connected to them. The only hidden connections on your script will be the ones between a Stamp and an Anchor.
8. It has been thoroughly **tested in production** by people coming from different studios, and its development has been shaped by all the different backgrounds and use-cases. Win/Mac/Linux.
9. It is **customizable** to each specific pipeline's requirements and naming conventions.
10. Nuke **scripts containing Stamps can be opened by anyone** that doesn't have them installed. So it can be also installed locally by single users without erroring on the farm, or even company-wide still being able to share the Nuke scripts with other studios as needed.
11. Using ***Stamps*** for the artist is as easy as **remembering a single hotkey**. The hotkey is context-dependent, so it performs many different actions based on what's selected.
12. It is **free**.

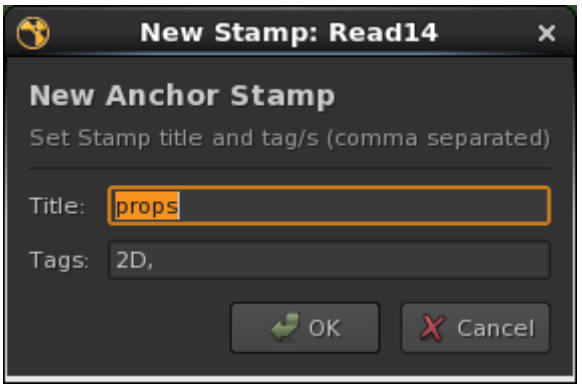
3. How to use

> ***Note:*** All you need to remember in order to use Stamps is the key `F8`, which is context-based, and depends on what's selected. You can change it for any other key or shortcut, but for the rest of the guide we'll use the default `F8` for simplicity.

3.1. Creating your first Stamp

To create your first **Stamp**, select a node on your Node Graph and press: `F8`

A pop-up will appear asking you for a `Title` and optional `Tags` for your **stamp**.



Title

The `Title` is the text that gets **shown in the Node Graph**. It is **related to the Stamp** and helps the user identify it later.

Please note that the `Title` doesn't correspond to the internal name of the **Stamps**, but it's only for display purposes. This has the following advantages:

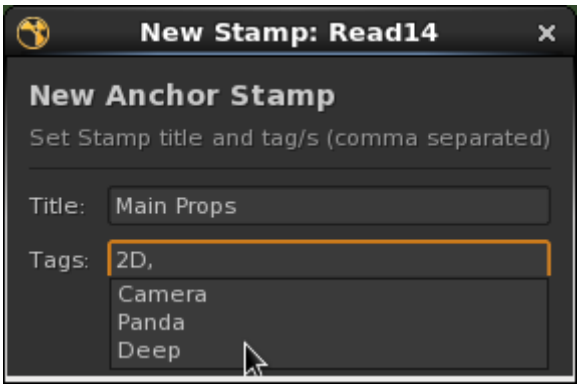
- Several nodes can display the same exact Title text on the Node Graph.
- You can even have multiple **Anchor Stamps** with the same title.
- You can use spaces or almost any character for the **Stamp's** `Title`.

> *By default, the panel will try to guess the best title for the stamp. In most cases this is just what you need. In general, this will be the node's name, but there are exceptions that you can extend or modify, like the following ones:*
>
> - *If the node has a `file` knob filled in, it will guess it from there. Like the example from the picture above, where it guessed **props**.*
> - *The default `Title` on the first Stamp made for a Camera will be just **cam**.*
>
> *These rules for the default titles save a lot of time in the long run while helping maintain consistency through the comp team, and can be custom written in Python. See the **Advanced** section for more on how to configure this.*

Tags

`Tags` are a quick way of categorizing the **Stamps** to easily find them afterwards. You can type as many tags as you want, comma-separated.

Every time you type in a comma, a dropdown completer will pop-up displaying all the other tags that already exist in your script. However, you can type any new tag/s.

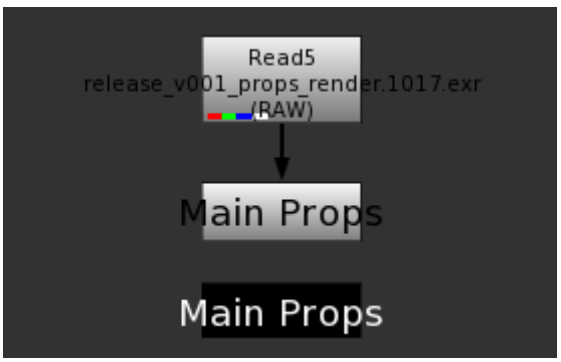


> *By default, the panel will add a tag based on the type of the node. We've found this alone helps a lot afterwards. When you create a Stamp for a Blur o Read node, the default tag will be **2D**. For a Sphere, it will be **3D**. For a Camera, **Camera**. Etc.*

> ***Tip:** If you select multiple nodes (for example the new FX renders) and press `F8` to create stamps for all of them, if you write an additional tag (for example **FX**) on the first one, it will transfer itself to the next panels too.*

3.2. Anchor and Wired Stamps

After pressing `OK`, we can see that two nodes have been created. We refer to them as **Anchor** and Wired **Stamp** (or just ***Stamp***).



Anchor

The white node, with the visible input, is our **Anchor** Stamp. An **Anchor**:

- Visually tells us which node our **stamps** point to.
- Stores the `Title` and `Tags`.
- Lets us interact with its dependent **stamps**.
- Serves as a visual reference to the only places where there can be hidden lines.

Stamp

The black node, with the input hidden, is our Wired **Stamp**, or simply **Stamp**.

- You can move a **stamp** anywhere on the script and start working with it.
- You can copy-paste it, and it will auto-reconnect every time to its **anchor**.
- It lets you interact with its **anchor** and with its similar **stamps** (siblings).
- It stores and displays on the Node Graph the same `Title` as its Anchor and siblings.
- It has smart reconnection features for all the different cases.

This is the main concept you'll need in order to start working with **Stamps**. We'll cover all the properties of the **anchor** and **stamps** later.

> ***Stamps** are built over default Nuke nodes, so a nuke script containing them can be opened by anyone that doesn't have them installed. Therefore it can also be installed locally by single users without erroring on the farm, or even company-wide while still being able to share the Nuke scripts with other studios as needed.*

3.3. Ways to create Wired Stamps

There can be as many **stamps** as you want for a single **anchor**. You can create them in the following ways:

- **By copy-pasting an existing stamp** (`ctrl+c`, `ctrl+v`)

  After doing so, the pasted **stamp** will immediately reconnect itself to its anchor. This also works when copying node trees that include multiple **stamps**.

- **By opening the Properties panel of an Anchor, and clicking on** `new`

  This will create a new **stamp** right below the selected **anchor**.

- **By selecting an existing stamp, and pressing** `F8`

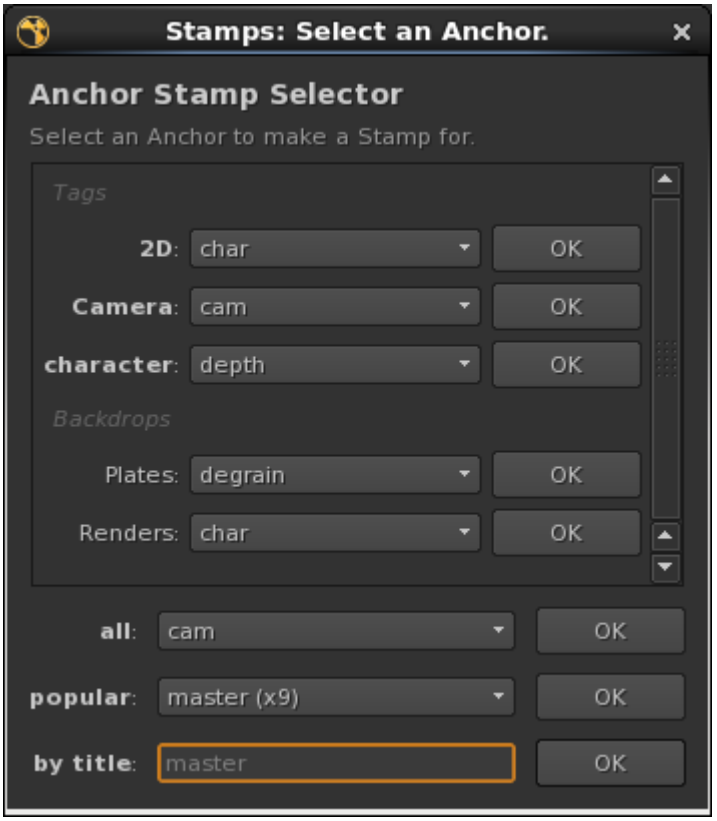
  This will create a new **stamp** next to it, as if you had copy-pasted it to the side.

- **By deselecting everything, and pressing** `F8`

  This will bring the **Anchor Selection Panel**.

3.4. Anchor Selection Panel

The **Anchor Selection Panel** if a quick and convenient way to create a **Stamp** in any part of your script, selecting from all the **anchors** organised by their tags and backdrop nodes.



It lets you filter through the **tags** and **titles** of the **anchors**, as well as the **labels** of the **Backdrop nodes** where they are.

To open it, click on the Node Graph, where you want to create your **stamp**, and press `F8`.

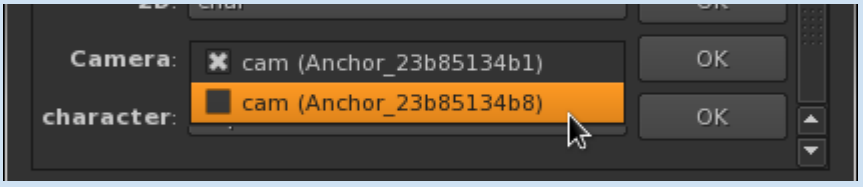
Then, select a **stamp** `Title` from any dropdown menu, or type it on the "**`by title:`**" text input.

Finally, press the `Enter` key, or click on the corresponding `OK` button. A **stamp** will be created at the desired position on the Node Graph.

Dropdowns

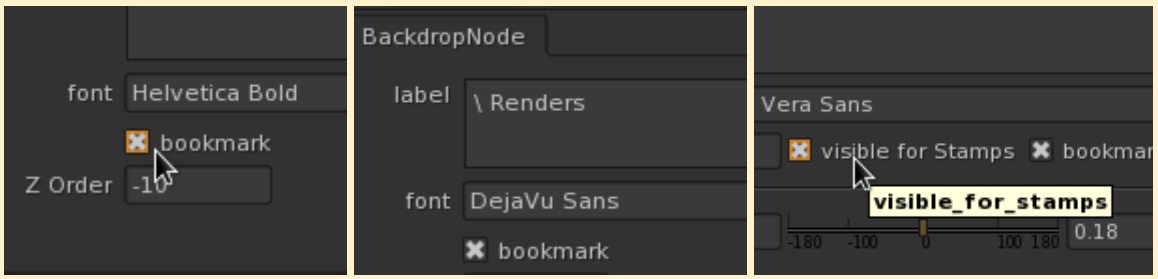
The **Anchor Selection Panel** has different dropdowns and `OK` buttons, that include **the titles of all the anchors**, sorted in different ways.

- One for every available **tag** it finds on all the **anchors**.
- One for every label of a **Backdrop** node that includes one or more **anchors** inside.
- **`all`**: Contains the titles (alphabetically sorted) for **all the existing anchors**.
- **`popular`**: Also contains **all the anchor titles**, but sorted by the amount of **Stamps** each **anchor** has connected to it.

> *Each dropdown contains the title for all the anchors that contain that tag.*
> *However, in the case where different **anchors share** at the same time the **title** and the same **dropdown**, they will be shown as:* `title (real node name):`
>
> 
>

***Tip:** If you want to **hide any Backdrop that contains stamps** from the `Backdrops` list on the `Anchor Selection Panel`, you have **three ways**:*

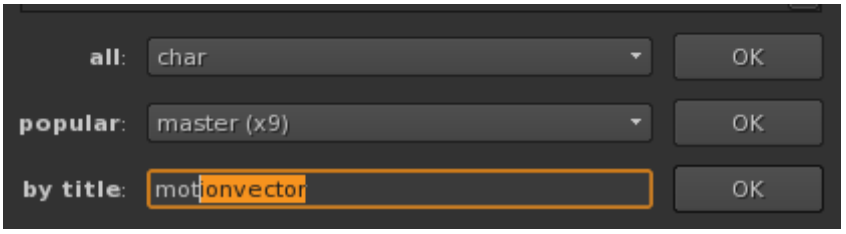
1. *By disabling the **`bookmark`** knob on the Backdrop node.*
2. *By adding a backwards slash (\ ) at the start of the label of the Backdrop node (it will not get shown on the Node Graph).*
3. *By having a checkbox knob named **visible_for_stamps** unchecked, on the Backdrop node. This is useful on custom Backdrops only.*



Text input

The **`by title:`** text input will auto-complete with predictions as soon as you start typing.

- Non-case sensitive.
- When you're happy with the predicted word, just press `Enter`.
- If you enter just a part of a `title`, it will pick the first **anchor** that it finds whose title starts with the entered text.
- In case multiple anchors share the same title, you'll also be able to include the name.



***Tip:** If you leave **`by title`**: blank, it will show a suggestion for the last **stamp** you created. So you can just press `Enter` or `OK`, and it will be created again. This is super convenient when you're creating the same **stamp** over and over.*
...just click on the Node Graph, press `F8`, and `Enter`:

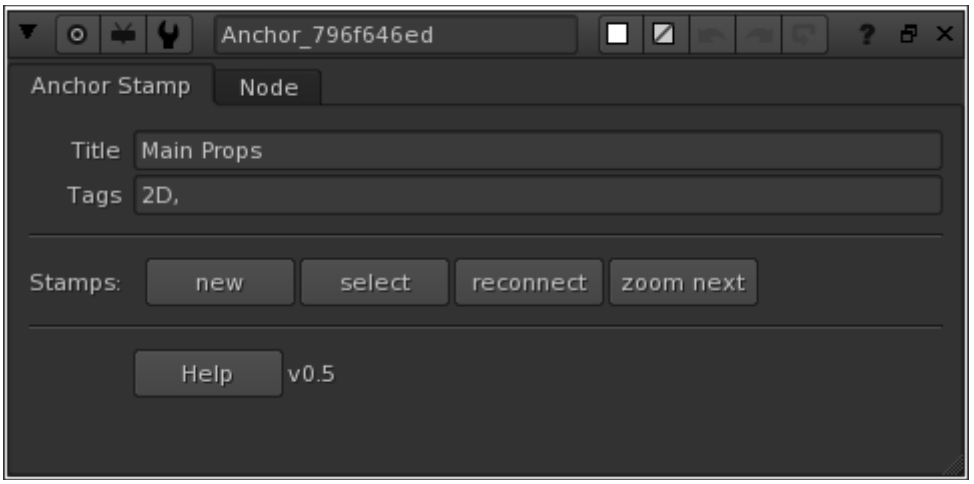
New in Stamps v1.1: A quick and rather hidden way of creating multiple stamps at once is by right-clicking the OK buttons as needed, and then finishing the selection with left click.

3.5. Controls on the Anchors and Wired Stamps

The **anchors** and **stamps** have some more controls exposed on each node's Properties panel. They let you rename the shown `title`, manually add or modify the `tags`, navigate through them or reconnect them in multiple ways.

Anchor Properties

This is what the **anchor** Properties panel looks like (double clicking on an **anchor**):



- `Title` and `Tags` can be modified directly.

  When modifying the `title`, it will ask for confirmation in order to apply it to the connected **stamps** too.

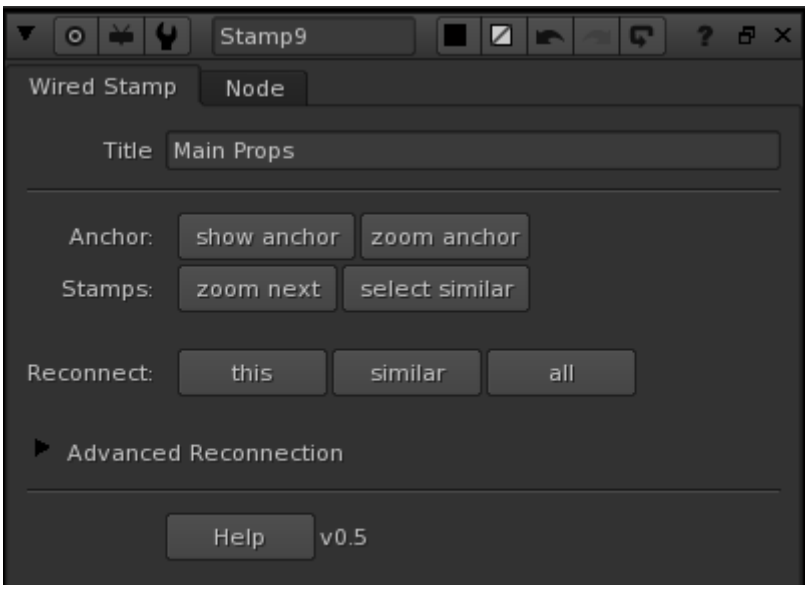
- Buttons on `Stamps:`

`new`	Create new **stamp** for this **anchor**, right below it.
`select`	Select all **stamps** connected to this **anchor**.
`reconnect`	Reconnect all **stamps** that refer to this **anchor**.
`zoom next`	Center the Node Graph's zoom to this anchor's next **stamp**.

- `Help` should open the documentation for **Stamps**.

Stamp Properties

Stamps also expose different buttons for navigation and reconnection, on their Properties.



- The `Title` can be modified directly. It will then ask for confirmation in order to apply the `title` changes to the **anchor** and its connected **stamps** too.

- Buttons on `Anchor/Stamps`:

`show anchor`	Open the Properties panel for this **stamp's anchor**.
`zoom anchor`	Navigate to this **stamp's anchor** on the Node Graph.
`zoom next`	Navigate to this **stamp's** next sibling on the Node Graph.
`select similar`	Select all the siblings of this **stamp** (useful to see them).

- The `Reconnect` buttons will try to reconnect **stamps** to their **anchors**, when they have been disconnected for some reason. The first three buttons, next to *Reconnect:*, will connect the **stamps** to their **original anchor/s**. That is, the anchor which has an internal name saved inside of each stamp, that is stored on the moment of creation.

`this`	Reconnect this **stamp** to its original **anchor**.
`similar`	Reconnect this **stamp** and its siblings to their original **anchor**.
`all`	Reconnect **all the stamps** in the script to their original **anchors**.

- For cases where you don't want to reconnect **stamps** to their original **anchor**, but for example to an **anchor** that shares the stamp's `title` (no matter its name), there are **Advanced Reconnection** options. We'll look into them on the next section.

4. Advanced

We now know the basics and are ready to use **Stamps**. However, they include several functions for reconnection, tagging, etc., and it is really convenient to understand how **stamps** work on the inside in order to make the best use of the tool.

In this section we will cover the insides of the **stamps** and the extra functionality.

4.1. Name and Title

Name

The internal **name** (node name) of the wired **stamps** is Stamp1, Stamp2, etc.

However, the internal **name** of the **anchors** is `Anchor_` followed by a 10 character hash, that is generated on creation of the node. Example: `Anchor_2d8e518c89`

> *The main advantage of this over Anchor1, Anchor2, etc. is that through a hash you ensure that no other node will be named the same way. This is helpful when copy-pasting stamps from one script to another, as if they belong to the same anchor (i.e. from a template), they will auto-reconnect straightaway.*

The way each stamp knows which one's its anchor, is through the anchor's name. It is stored in a knob named `anchor`, inside of each **stamp**, under *Advanced Reconnection*:

This is the name that the **stamp** will always look for when finding its **anchor**, both for auto-reconnection on copy-pasting and for normal reconnecting via the *Reconnect* buttons.

> ***Note:** When you copy-paste an **anchor** inside the same nuke script, as its name exists, it will have to get a new one. This won't break anything, and any **stamps** you copy alongside with it will also pick up the new anchor name on creation. So even if they have the same `title`, each **stamp** will know which one is its **anchor**.*

Title

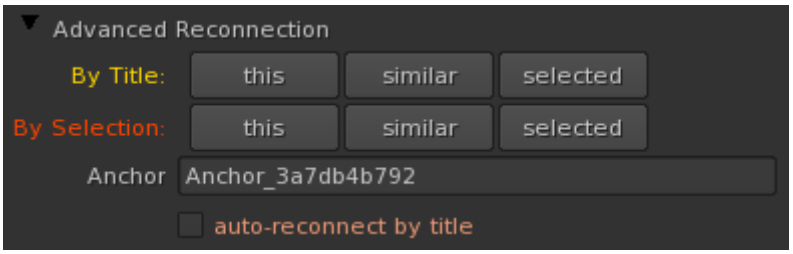
It is the text that the **anchor** and **stamps** display on the Node Graph, and not their real name (which is usually the one displayed on the Node Graph with normal nodes). As shown earlier, this has multiple advantages for the needs of **stamps**.

The **title** is stored in an editable knob named `title`, on the **anchor** as well as on each individual **stamp**. When you modify it, either on an **anchor** or on a **stamp**, it will ask you for confirmation, and then apply the new title on all the connected stamps/anchor.

4.2. Advanced Reconnections

As mentioned earlier, when you reconnect your **stamp**, most times you'll want it to connect to the **anchor** whose name it has stored in its `anchor` knob. However, there are situations where this is not enough or applicable, so **stamps** offer a few extra ***advanced reconnection*** methods that we can see now that we understand how the name and title are stored.

You can access these functions through buttons on any **stamp**'s Properties panel.



Reconnect by Title

The **stamp** will try to find an **anchor** which shares its ***title***, even if its internal name is not the one stored on the `Anchor` knob.

`this`	Reconnect this **stamp** to the **anchor** that shares its title.
`similar`	Reconnect this **stamp** and its siblings to the **anchor** that shares their title.
`selected`	Reconnect **all selected stamps** to the **anchor** that shares each one's title.

> ***Tip:*** *This is especially useful when bringing nodes from one script to another. Imagine both scripts have an **anchor** titled "**cam**", but they were created at different moments, so they have a different internal name (Anchor_17478e7b0f and Anchor_c837ae11c9). However, when you copy-paste **stamps** from one script to the other, you'd like them to connect to the anchor titled "**cam**", whatever its name. Here's **Reconnect by Title**.*

> *If there are multiple **anchors** with the same **title**, a pop-up will appear asking you to select the **anchor** you want, and then click the button again.*

Reconnect by Selection

This is the ***brute-force*** reconnection, which can be really useful when you want to manually force one or more **stamps** to belong to a different **anchor**. In order to reconnect by selection, you must first select one **anchor** (and only one) on the Node Graph.

`this`	Reconnect this **stamp** to the selected **anchor** on the Node Graph.
`similar`	Reconnect this **stamp** and its siblings to the selected **anchor**.
`selected`	Reconnect **all selected stamps** to the selected **anchor** (there can only be one anchor in the selection).

> ***Tip:*** *Reconnect by selection will change the **title** and **anchor** of the stamps to the ones of the new **anchor**. An example where this is especially useful is if you find yourself having three anchors named **cam**, **mainCam** and **Camera**, because they belong to templates you took from different places and the artists weren't consistent setting them up. You could select all the **stamps** you want (even including non-stamp nodes in the selection, it will ignore it) as well as one of the **anchors**, and **reconnect selected by selection**. Now they will all belong to a single anchor and you can kill the other two.*

Auto-reconnect by title

The checkbox in the end named "auto-reconnect by title" **when enabled** does the following:

The next time this specific stamp is created (this is, when you copy-paste it or bring it from a template), it will auto-reconnect by its title instead of by its anchor name, and then, the checkbox will be auto-turned off immediately. This is **really useful for templates**, or any time you want **stamps** to auto-reconnect ignoring the internal name when you bring them.

> ***Tip:*** *This works especially nice with naming conventions, either in a studio or for yourself. For example, if you always title the main camera **cam**, you could make a stamp titled **cam**, activate its "auto-reconnect by title" checkbox, and connect it for example to a ScanlineRender and save both nodes as a template/toolset. Now, whenever you bring that toolset in any nuke script, if you have an anchor titled **cam**, the stamp from the toolset will connect to it straightaway, even if its internal anchor name (the one with the hash) is not the same because they were created in different places.*

> ***Note:*** *Since **Nuke 11.3v1**, there is this annoying **bug/feature** where nodes that are connected through a callback get a green expression arrow, as if they were expression-linked (which they aren't). I hope this will be resolved soon in future updates of Nuke. However, it **won't cause any problems apart from the visual discomfort**.*
> *Also, it is only visible while selecting the stamp, and **disappears by itself** as soon as you reopen the script. You can also get rid of it by clicking on any Reconnect or Refresh button.*

4.3. Smart behavior (callbacks and style)

We will now cover a bit more in-depth the smart behavior of the **stamps**, so that even if it should be predictable enough, you can troubleshoot better and know what can be going wrong and what not.

> *This is a bit **technical** and requires understanding of how Nuke callbacks work. And it's **mainly for troubleshooting** purposes, so feel free to skip to the next chapter if you don't need it right now.*

Anchor creation

When an **anchor** is created, it gets assigned a hash name (Anchor_f72c3c56ad) and it doesn't change from that moment. It also gets assigned a title which can be changed later.

Stamp creation via the panels or shortcut

When a wired **stamp** is created through the panels or through the shortcut, the first thing it does is take its anchor's name and title and store them inside the knobs `anchor` and `title`. The `anchor` knob should never be edited manually. The `title` knob can be modified.

Stamp creation via copy-pasting: auto-reconnection

The way the auto-reconnection happens is the following:

1. First, on copy-pasting, the **onCreate** callback of the stamp gets activated. This turns a hidden checkbox knob called ***toReconnect*** into `True` (only if the GUI is running).
2. The **knobChanged** callback on the stamp checks every time if the value of its knob ***toReconnect*** is `True` (only while the GUI is running). Once it's True, it will immediately turn it `False` and then perform the reconnection to its anchor based on the name saved on the anchor knob.
3. If the reconnection is not successful, it will change the style of the node's font, to red color and double size. This will revert as soon as you reconnect the **stamp**.

> *If on creation, a **stamp** is already connected to an **anchor** sharing its **title**, it will change its anchor **name** to the new one without asking. What this means, is that **if you copy-paste anchors and stamps together, the newly pasted stamps will stay connected to the newly pasted anchors.***

4.4. More on Tags. Add and rename tags.

Tags are one of the core concepts of the **stamps** workflow. They make a big difference in the speed at which you access your nodes from any part of the script, many times per day.

Default tags

When you're creating new **anchors**, the ***Anchor Creation Panel*** gets some **tags** by default:

- **Node type tag**
  This can be **2D, 3D, Camera, Deep** or **Particle**.

  > *If the **stamp** is created on a **Dot**, or **Stamp**, it will look for the type and other information upstream, recursively.*

- **Backdrop labels**
  When a node where you create a **stamp** is inside a Backdrop, if the Backdrop has a text on its label, the text from its first line will be included as a default tag.

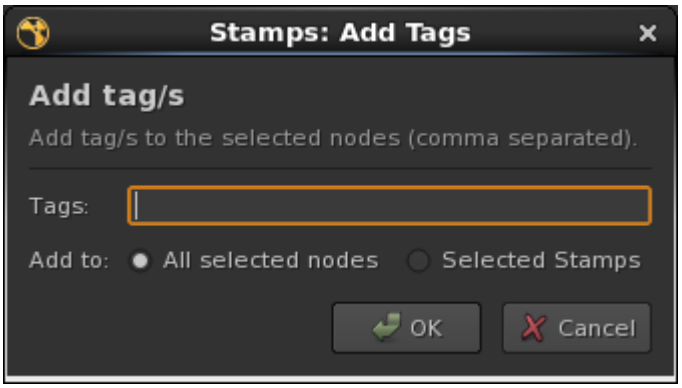
  > - *Any html entities like will be omitted.*
  > - *If there are multiple backdrops inside each other, their tags will be added.*

- **Custom default tags**
  There is an option for leads or TD's to define a custom python function that generates custom default tags based on the input node. This will be discussed in the next chapter, ***4.5. Customization***.

- **The stamp_tags knob**
  If the node where you create a **stamp** has a text knob named `stamp_tags`, the default tags for the **stamp** will only be the text of the `stamp_tags` knob. Again, this is useful for the creation of templates.

You can add as many tags as you like, comma-separated, in the tags knob on the **anchors**. However, to edit multiple anchors at the same time, there are two convenient panels:

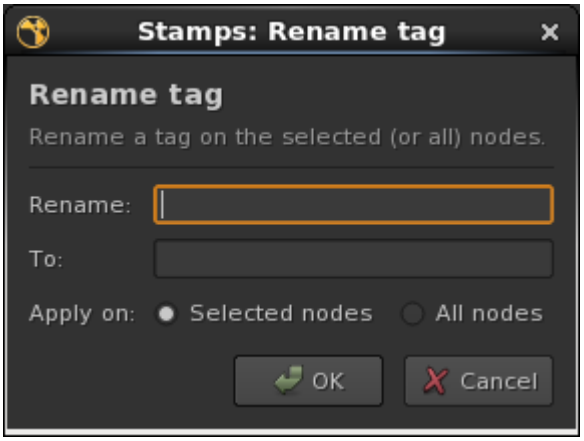
Add tag/s to selected nodes `(Edit/Stamps/Add tags to selected nodes)`



This lets you add one or more tags (comma-separated) to the selected ***stamps***, or to ***all the selected nodes***.

If you choose ***All selected nodes***, any non-anchor nodes will get the knob `stamp_tags` created on them, including your added tags.

Rename tag `(Edit/Stamps/Rename tag)`



This lets you rename a tag on the selected ***nodes***, or on ***all nodes***.

Both the *`Rename:`* and *`To:`* text input knobs have an auto-completer as soon as you start typing, with all the available tags in the nuke script.

Please note that the tags are case sensitive.

4.5. Customization.

> ***This section is only useful for TDs or the people setting up stamps.***

Included with **Stamps** there is a python file named **`stamps_config.py`**. This file lets you modify different aspects of the Stamp style and defaults, while **letting you simply replace stamps.py every time there is an update**.

All you need to do is copy **`stamps_config.py`** in your nuke plugin path. For example, right next to **`stamps.py`**. This is completely optional, as stamps will still work without that file.

Although **the code is self-explanatory and even includes some examples**, we will now see the main functions and constants that it makes available to us:

Main Defaults: `STAMPS_SHORTCUT, ANCHOR_STYLE, STAMP_STYLE`

These define the custom shortcut (defaults to `F8`), and the UI colors of the stamps.

Custom Functions: `defaultTitle(node), defaultTags(node)`

You can use these to override the default title and tags that get assigned on the creation of a new anchor on any given node. If you return `None`, the default-default one/s will take over.

Advanced: `KEEP_ORIGINAL_TAGS, Exception classes, StampClassesAlt`

`KEEP_ORIGINAL_TAGS`	Append the normal defaults to `defaultTags()` or not.
`Exception classes`	These define which node classes classify as which node type.
`AnchorClassesAlt` `StampClassesAlt`	Override certain node types to get their stamp as a different class (instead of NoOp). This lets you have different shapes, i.e. using DeepExpression for Deep nodes or Axis for Axis.

4.5. Available python functions.

> ***This section is only useful for TDs or the people setting up stamps.***

Some of the internal python functions that **Stamps** uses might be useful to use directly from nuke. The way to run them will be as follows:

```
import stamps; stamps.functionNameHere()
```

Some available functions:

- `stampCreateByTitle(title="")`
  Given a title, tries to create a new stamp connected to the first anchor with that title.
- `allAnchors(selection="")`
  Returns a list of all anchors within the nodes selection. If empty, returns all anchors.
- `anchor(title,tags,input_node,node_type="2D")`
  Creates a new anchor, given its title (str), tags (str), input_node (node) and type (str).
- `wired(anchor)`
  Creates a new wired stamp, given the anchor node.
- `findBackdrops(node)`
  Returns a list of backdrops that have the node inside of them.
- `realInput(node)`
  Looks upstream recursively and returns the first input that is not a Dot or Stamp.
- `allAnchors(selection="")`
  Returns a list of all available tags within the selection. If empty, within the full script.
- `toNoOp(node)`
  Converts a stamp of any class into a NoOp stamp.
- `allToNoOp()`
  Converts all stamps into NoOp stamps.

> ***Tip:** You can also find **other useful functions** inside the python buttons in the **stamps**.*

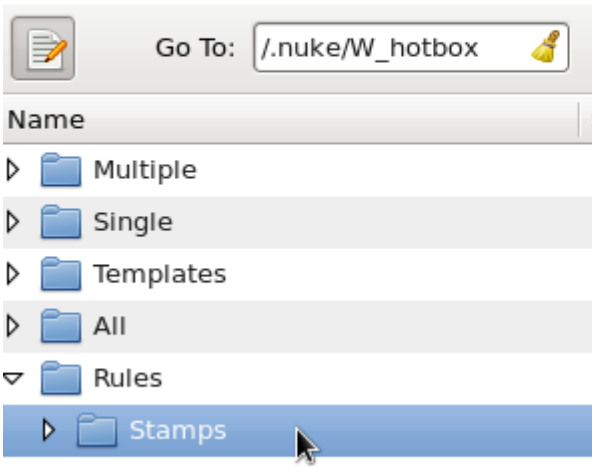
5. Extras

This section is about the ***extra*** stuff that either comes with **Stamps** or might be useful for its workflow. It will be expanded with future updates.

5.1. W_hotbox integration

For anyone using **W_hotbox**, the **Stamps** package includes a group of convenient buttons for it, that only show when selecting **stamps**.

You can **download W_hotbox** here: http://www.nukepedia.com/python/ui/w_hotbox



In order to **install the Stamps rules for W_hotbox**:

Simply copy the included folder:
`W_hotbox/Rules/Stamps`

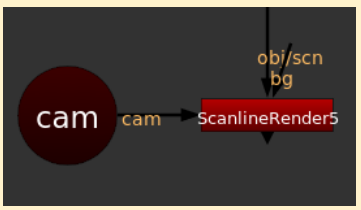
...inside your:
`/W_hotbox/Rules/`

Which probably lives inside
`~/.nuke`

5.2. DummyCam

DummyCam is a smart Camera that apart from the matrices also grabs all the "Projection" values from the upstream camera that it's connected to. This happens live and super fast through TCL. **DummyCam** is included with **stamps**, and also was released individually.

> ***Tip:*** *Combined with **stamps**, **DummyCam** lets you have the **Camera stamps** shaped like a Camera. All you need is to have* `DummyCam.gizmo` *in your plugin path (i.e. next to stamps.py), and the Camera stamps will look like this:*



6. Installation

A. Recommended method

1. Copy the *stamps* folder and paste it anywhere in your plugin path (like in *~/.nuke*).
2. Open the file *init.py* inside your *~/.nuke* folder with a text editor, or create it if it doesn't exist.
3. Add the following lines:
   ```
   import nuke
   nuke.pluginAddPath("stamps")
   ```
4. Restart nuke.

B. Manual method

1. Copy the desired files, like *stamps.py* or *stamps_config.py* somewhere in your plugin path.
2. Find or create *menu.py* , and append the following: `import stamps`
3. Restart nuke. Please be aware this won't load the includes (like W_Hotbox buttons or DummyCam).

7. Contact

Happy to help with:

- **Feedback** and suggestions.
- **Bugs** and **feature requests**.
- **Problems** or **assistance** getting **Stamps running on a pipeline**.

You can just drop an email at pueyovfx@gmail.com

8. Special Thanks

We wanted to finish this guide by showing our gratitude to the people that helped through the development of Stamps, by giving awesome ideas or feedback or by using it in an unfinished state it in different productions for the sake of beta testing.

Especially to Ernest Dios for expanding over some of the concepts of the tool. Tony Lyons for nice ideas and coming up with the logo concept. Also, to Trixter Munich for being the first studio to implement the production-ready version of Stamps.

9. Updates Log

Stamps v1.1.0 *- 17 May 2021*

New features:

1. Postage Stamp thumbnail option for the node graph on “Type 2D” Stamps.
2. Anchor Stamp Selector: multi-select by right clicking the OK buttons before a left click.

Bug fixes and enhancements:

1. Compatible with Nuke 13 / Python 3.
2. Anchor Stamp Selector: “Popular” dropdown behaviour fixed.
3. DummyCam updated to v1.2.

10. License

www.adrianpueyo.com

User guide updated:

May 18, 2021