

# Assign materials

## Shader Assignments

A gpuCache is a single shape in Maya, so it is impossible to assign materials as usual. But we can use RenderMan's **Dynamic Rules Editor** to assign materials at render-time, with the following limitation:

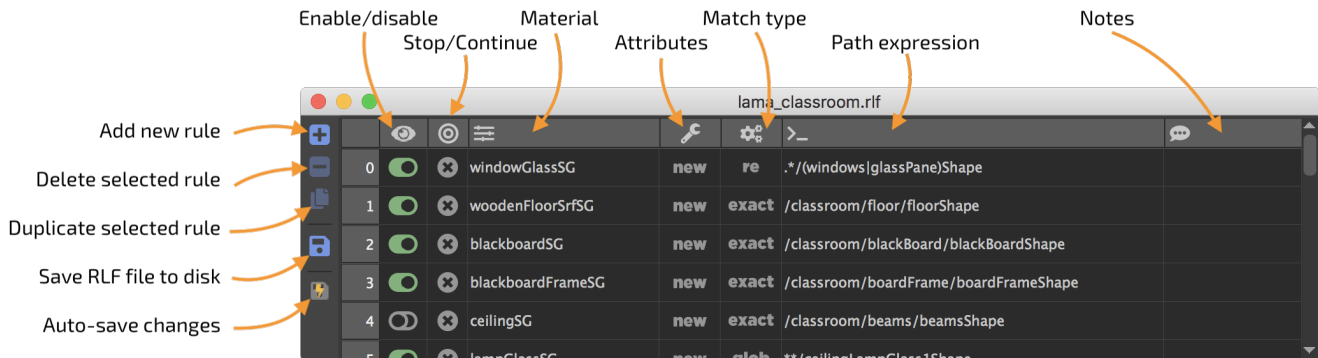
- All referenced materials must be present in the scene.
- Per-faceset materials are not supported.
- Any attribute influencing the raytracing core will be ignored by the renderer. This includes group membership definitions, selective lighting, etc.
- Any rule change will require an IPR restart to take effect.
- These limitations will be lifted in an upcoming release.

- [Shader Assignments](#)
- [RLF Editor overview](#)
  - [Rules management](#)
  - [Rules order](#)
  - [Rule controls](#)
  - [About attributes and node types](#)
- [In-scene shading](#)
  - [Assign a base material to part of an archive](#)
  - [Match sub-strings in a hierarchy](#)
  - [Rule execution order](#)
- [Self-contained Asset Shading](#)
  - [Notes about RLF export](#)
- [Dynamic Rules CookBook](#)
  - Everything in the scope, i.e. the Maya scene:
  - All nodes in the gpuCache node:
  - All nodes of the gpuCache with "rivet" in their name:
  - All shapes of the gpuCache under a transform starting with "ring\_":

## RLF Editor overview

Open the Dynamic Rules Editor and let's see how it works.

You can either click the double angle bracket icon in the shelf or the **Edit RLF File** button in the gpuCache's attribute editor. The shelf button edits all scene alembics and the per-node button edits only the RLF associated to that node.



## Rules management

The left-side buttons handle all rule and file management tasks:

- **Add new rule:** create a new empty rule
- **Delete selected rule:** Select one or more rules by clicking the rule number and press the button.
- **Duplicate selected rule:** Select one or more rules by clicking the rule number and press the button.
- **Load RLF file:** replace the editor's contents with a file on disk. (scene editor only)
- **Save RLF file:** save the editor's contents to disk.
- **Auto-save changes:** automatically save to disk when the icon is orange. (node editor only)

## Rules order

Rules are evaluated in the editor's order. To change their order, click in the rule number and drag to the new position.

## Rule controls

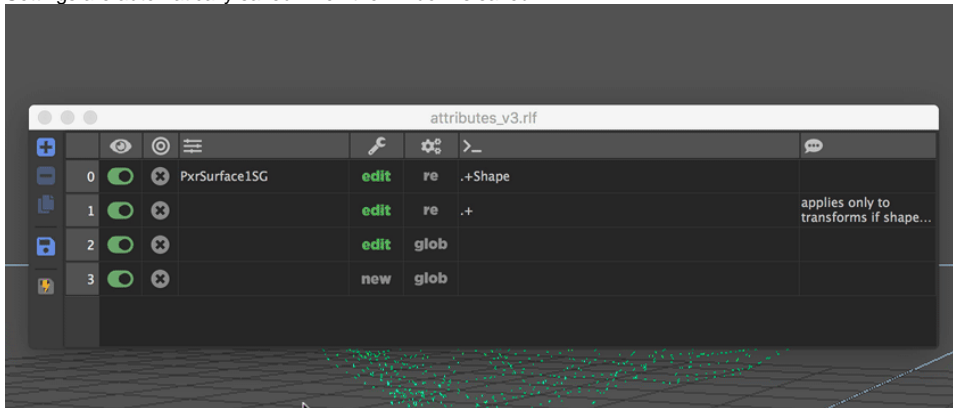
**Enable/disable:** when the toggle is green the rule will be evaluated.

**Stop/continue:** When the icon is a cross, the rule evaluation will stop if this rule is matching. The arrow down icon indicate that subsequent rules will be evaluated even if this rule is matching.

**Material:** The shading group of the material, which references the surface and displacement shaders.

**Attributes:** insert or override attribute values. Click the **new/edit** icon to open the attribute dialog.


- Click the + icon and right-click the attribute field to choose one of the supported attributes.
- Select one or more attributes as usual and click the - icon to delete them.
- Attributes with a fixed list of acceptable values can be set by right-clicking too.
- Settings are automatically saved when the window is saved.




**Match type:** There are 3 matching engines:

- **exact:** this is the fastest and will only match a full path.
- **glob:** based on glob matching with the following tokens:
  - \* : any number of character
  - ? : a single character
  - \*\* : any depth in the hierarchy
- **re:** [re2](#) regular expression. The most sophisticated, complex and potentially slow matching method.

**Path expression:** You can either double-click in the field to type your expression or drag and drop from the alembic node list.

 To drag and drop, you must first select the node in the alembic node list.

**Notes:** A field to write notes when things get complicated...


 A rule can assign:

- a material
- attributes
- or both !

## About attributes and node types

In RenderMan, every shape is an instance. This means that an attribute attached to a specific shape will influence all instances of that shape.

If you want to assign an attribute to an instance, create an expression matching the instance's transform. This is particularly important for classic attributes like `maxspeculardepth`, `maxdiffusedepth`, `relativepixelvariance`, etc.

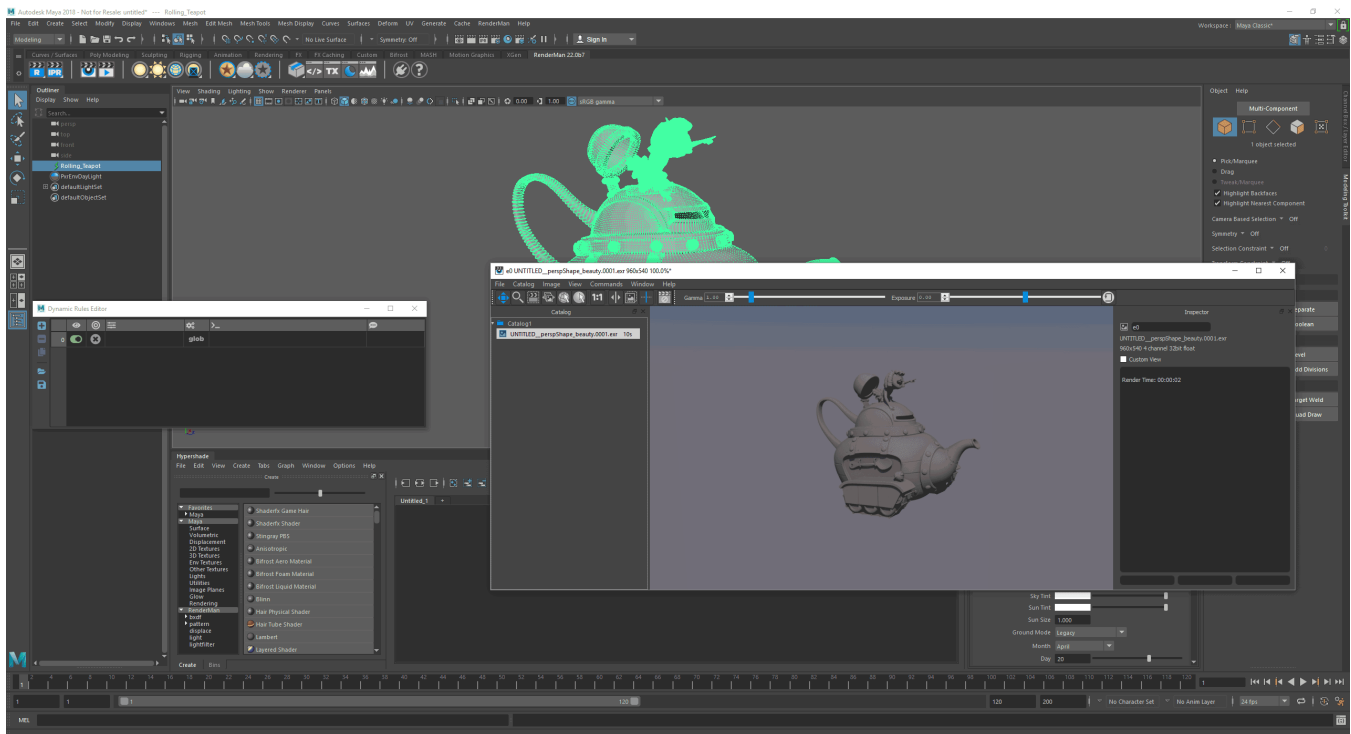
 **Upgrading RenderMan 22.x RLF files 23.x:** the RLF editor must be opened and the Alembic Cache contents need to be read into RFM in the AlembicCache node. This data must be brought into RenderMan for Maya *before* rendering, if not, it won't upgrade the version automatically and will fail to render the RLF data.

Essentially, the RFL editor must be opened before rendering to allow it to recognize and upgrade the RLF file.

## In-scene shading

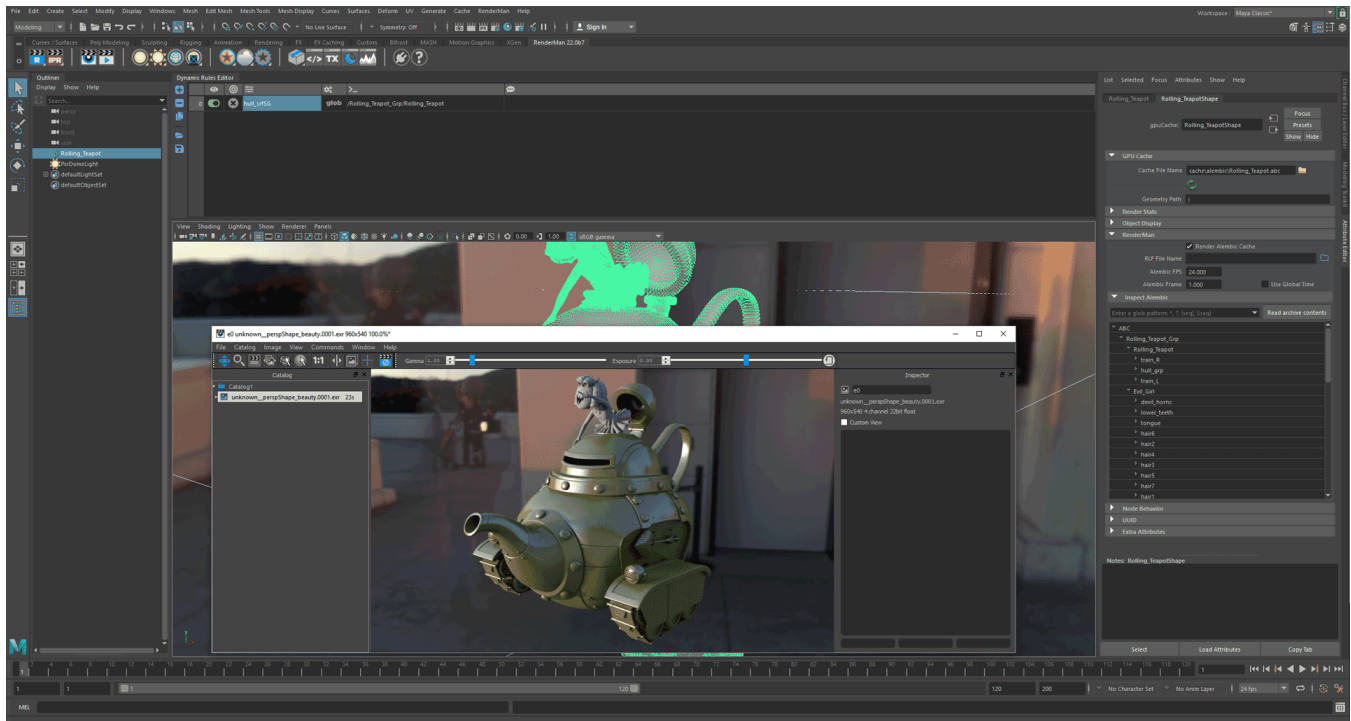
Dynamic rules are written using glob expressions, similar to a shell. See the [Dynamic Rules CookBook](#) below for more info.

This is our starting point: the gpuCache node renders with the default material:



## Assign a base material to part of an archive

We have prepared a number of basic materials to assign. Let's start by assigning the `hull_srf` material to the whole teapot and write our first rule:



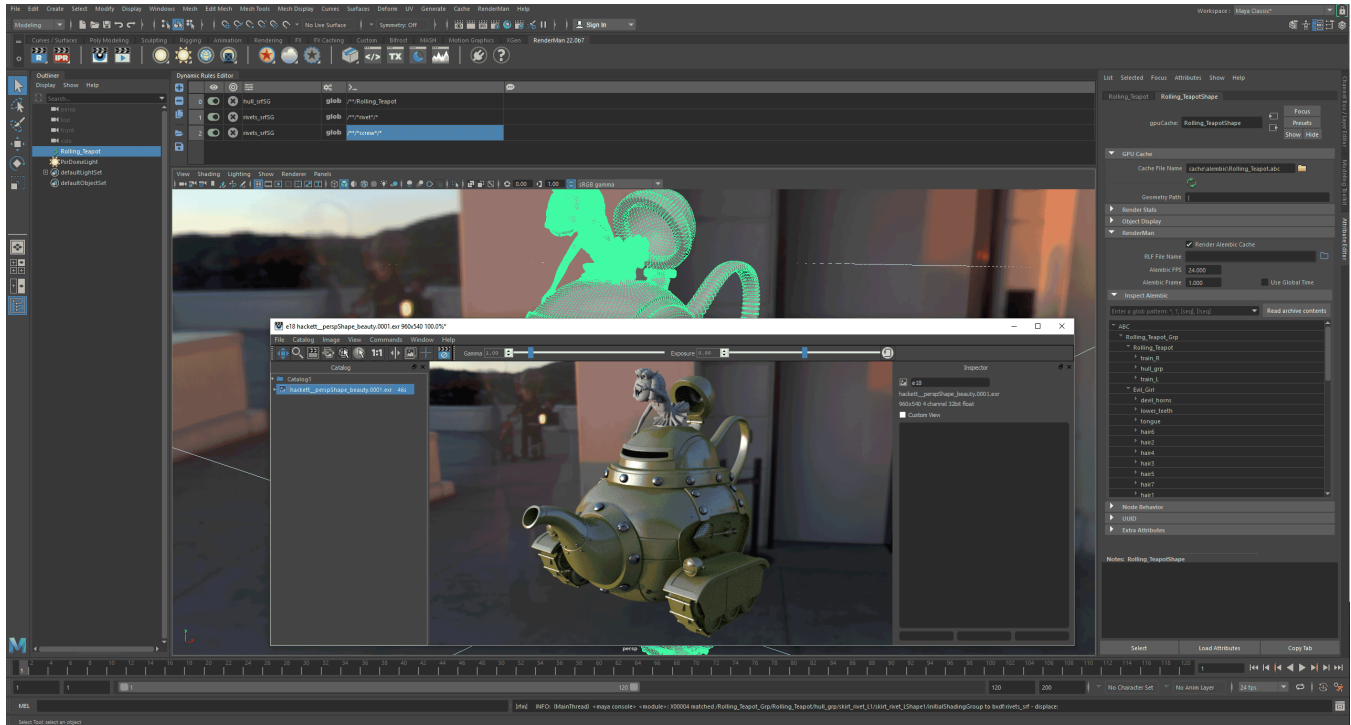
- In the node list, select the top node of the hierarchy we want to assign to: `Rolling_Teapot`.
- Click **Add Rule** (the blue + icon) to create a new rule or use the initial empty rule
- Double-click the expression field to put it in edit mode. This field is the `>` column.
- Go to the Alembic node list and middle-mouse-drag the selected node into the expression field.
  - Please note that you must select an item in the list before drag-and-dropping. (Your windows must also be docked and not floating on the Windows OS as the switching focus will not allow you to middle mouse drag.)
- Edit the expression from this `/Rolling_Teapot_Grp/Rolling_Teapot` to the below:

```
/**/Rolling_Teapot
```

- \*\* means match any depth of nesting (including none) whereas \* will only match characters between the / separator
- Press enter to validate.
- Right-click the **Payload** field (with the icon of sliders) and select `hull_srfSG`.
- Launch the IPR to test.

## Match sub-strings in a hierarchy

Now we will shade the rivets on the teapot. We will need a more complicated expression.



- Duplicate the first rule
  - Right-click on the rule number (0 in our case) and select **Duplicate Rules** (the copy icon just below the "-" minus icon).
- Extend the previous expression to match any object containing the word "rivet".

```
/**/*rivet*/
```

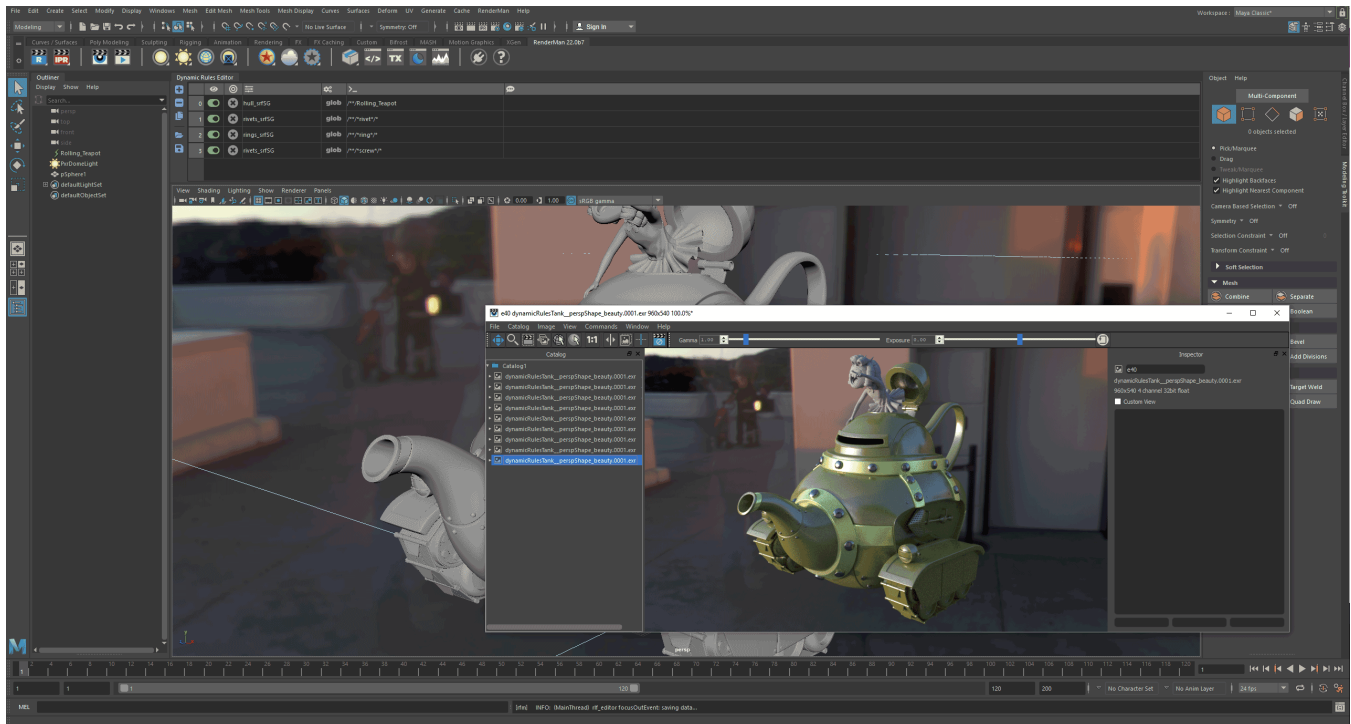
- The second \*\* means the rivet object can be nested at any level under the teapot
- \* rivet \* matches if rivet appears in node name. "rivet", "Arivet" and "Brivet2" would all match.
- On that model, there are also screws that could use the same material. Copy the expression to match those too.

```
/**/*screw*/
```

## Rule execution order

Let's assign a material to the teapot's rings.

We use the same workflow: duplicate the rule, replace "rivet" with "ring" and set the Payload to `ring_srf`. But it doesn't quite work because, as you can see in the node list, some shapes have both "rivet" and "ring" in their name and rivets on the rings are now using the wrong material.



We can change the **rule execution order** if the **result is incorrect** (maybe your order is different than mine in the image): if we match rivet first, the matching engine will stop when it finds a match and then only shapes only containing "ring" will match the second rule.

- Position the cursor over the rule's number and left-button-drag rule 1 (rivets) above rule 0 (ring).
- Render to check.

Of course, we have only scratched the surface and we recommend that you read up on [Regular Expressions](#) for more sophisticated use.

If you're done shading, select the disk/save RLF (RenderMan Look File) icon. This can be used in the import of the asset to assign the shading nodes each time. This can then be part of your asset publishing scheme.

## Self-contained Asset Shading

If you want to import fully shaded assets, there are 2 options:

- When RfM finds a RLF file in the same location and with the same name as the archive, it will automatically apply it.
  - This is not always very flexible in a pipeline, but may be useful for archiving assets.



The RLF file format does not include material descriptions and you have to make sure that:

- All referenced materials exist in your scene.
- **Output All Shaders** is checked in the Advanced section of the render globals.

- If the **RLF File Name** field of a gpuCache node contains a path to a RLF file, RfM will apply it to that gpuCache node.
  - It becomes easy to script look assignment and updating.



Note that you can still override automatic RLF assignments in the scene with the Dynamic Rule Editor !

In that case, the scene RLF rules will be executed before the archive's rules and thus have a chance to override them.

## Notes about RLF export

When you reference multiple RFL files in a scene, you must make sure that all shading nodes have unique names. If you define the same name multiple times, the renderer will use the first in line and you will get unexpected results.

You can export a RLF file by clicking the  icon in the RLF Editor, or write a simple Python script that will save the scene's current RLF data to a file:

```

import rfm2.api.nodes as apinodes

def save_scene_rlf(filepath):
    """Load the rlf data from the rmanGlobals node and save it to a file.

    Arguments:
        filepath {str} -- Full path to the final RLF file
    """
    rg = apinodes.rman_globals()
    if rg:
        json_str = mc.getAttr('%s.rlfData' % rg)
        try:
            with open(filepath, 'w') as fhdl:
                fhdl.write(data)
        except Exception, err:
            print 'Failed to write %s: %s' % (filepath, err)
    else:
        print 'Warning: Could not find rmanGlobals node in the scene !'

```

## Dynamic Rules CookBook

Imagine we have a gpuCache node named Rolling\_Teapot...

Everything in the scope, i.e. the Maya scene:

```
/**/*
```

All nodes in the gpuCache node:

```
/**/Rolling_Teapot
```

All nodes of the gpuCache with "rivet" in their name:

```
/**/*rivet*/
```

All shapes of the gpuCache under a transform starting with "ring\_":

```
/**/ring_*/
```