Installing Custom Nodes

In production, it's desirable to create and add your own patterns and C++ plugins (materials, lights, patterns, and more) to the user interface in Maya, having them load when RenderMan is loaded. This gives artists access to these nodes in a Maya session. Users wanting to do this must obtain a Maya Node ID from Autodesk (these are often assigned in blocks). This allows Maya to assign unique IDs and identify nodes.



Since we do not recommend changing the shipped rfm.json files, you may use a system environment variable, RFM_SHOW_PATH or RFM_SITE_PATH, to source your custom versions.

RenderMan for Maya will search in the following order and merge/override in the same order. **Note** : This hierarchical process is valid for the other config files: shelf.json, aovs.json, menu.json, etc.

1. rfm will read RFMTREE/config/rfm.json. This is the base configuration and should NEVER be altered by you, please make a custom version.
2. rfm looks for RFM_SITE_PATH and RFM_SHOW_PATH. These can either be environment variables or values in the main rfm.json (prefer the Environment Variable route).
3. rfm looks for RFM_SITE_PATH/config/rfm.json. if found, its contents will be merged with the base config.
4. rfm looks for RFM_SHOW_PATH/config/rfm.json. if found, its contents will be merged/override the base config and the site config.
5. rfm looks for ~/rfm/config/rfm.json. if found, its contents will be merged/override the base config, the site and show config. On Windows this would be Documents\config\rfm.json.

override.json can be used to override default values of RfM nodes both standard and custom.

To add extra nodes (patterns, bxdfs, integrators, etc), resources/user/nodes/search_paths should be extended. rfm always expects the same directory structure:

```
BASE
  |_ config
     |_ rfm.json
     |_ shelf.json, etc    (optional)
  |_ nodes    (optional)
     |_ args    (optional)
     |_ osl    (optional)
  |_ icons    (optional)
  |_ logs    (optional)
```

OSL Shaders

Node IDs for Patterns are embedded in the metadata. Below is an example from the PxrLayer OSL shader.

```
shader PxrLayer
   [[int rfm_nodeid=1053299, string
rfm_classification="rendernode/RenderMan/pattern",
    int rfc_nodeid=1037674, string rfc_description="Xpxrlayer",
    string help = "An OSL pattern for layerable parameters for PxrSurface or "
            "input layer to PxrLayerMixer."
   ]]
```

C++ Plug-ins

In your custom rfm.json file, you can load your files like the below example:

```
{
    "$schema": "./rfmSchema.json",
    "resources": {
        "user": {
            "nodes": {
                "node_exclusion_list": [
                    "PxrHair",
                    "PxrLMDiffuse",
                    "PxrLMGlass",
                    "PxrLMMetal",
                    "PxrLMPlastic",
                    "PxrLMSubsurface",
                    "PxrSkin",
                    "PxrValidateBxdf"
                ]
            }
        }
    },
    "dirmaps": {
        "user_to_z": {
            "from": "/User/name/maya",
            "to": "Z:/maya",
            "zone": "UNC"
        }
    }
}
```

Node IDs for these are included in the .args file. The order doesn't matter but it must be in the first level of the tags. The below example is for a light.

```
<rfmdata nodeid="1053261"
             classification="rendernode/RenderMan/shader/light"/>
```