Configuration

Overview

Configuration of RenderMan diagnostics primarily affects the presentation and reporting of data. There is no need to configure the gathering of stats; it is built into the renderer and cannot be disabled. The presentation of stats, however, is configurable through the use of an INI-style configuration file.

Live Stats

Live stats are enabled by default, which means that the system will automatically set up an internal server (the Live Stats Server) for reporting live stats for any running render. The `it` tool displays the tracked metrics as part of its HUD, and each DCC plugin also has a built-in "Live Stats" presentation panel. The Live Stats Server can be toggled on/off through prefs or the configuration file but the set of metrics collected by the system is currently not configurable.

⚠ Stats presentation for live interaction is enabled in RenderMan by default. The ability to attach to a render running on a farm is a feature planned for the future but not currently supported. The live stats system will spawn a background process `sbrokerd` which can interfere with process lifetime and cgroup management of farm processes therefore it is recommended that live stats be explicitly disabled for farm renders using a bespoke pipeline configuration file.

Configuration File

Default Configuration

The stats configuration file (default name: "stats.ini") holds the basic default settings for a stats session. Below is the default configuration for the stats system, equivalent to the default behavior of the system if there were no configuration file specified:

Default: stats.ini

```
# stats.ini
# Roz Stats default configuration file
#
# Copy this file to: /a/path/of/your/choosing/stats.ini
# set RMAN_STATS_CONFIG_PATH = /a/path/of/your/choosing

version 0.1

# Set this to 1 for extra information when parsing this file
verbose 0

# Stats processing log level.
# Range is 0 (none) to 5 (debug output). Default is 3 (warnings)
logLevel 3

# Session configuration
[Session]

# Distinguishing name for this session configuration
name "RMANTREE Session"

# Enable the internal live stats server
liveStatsEnabled 1

# List of listeners which the session should create and manage.
#[ManagedListeners]
```

This file can be found in the RenderMan distribution at `$RMANTREE/etc/stats.ini`

Config File Search Order

Command-line `prman` will search for the stats configuration file in the following order:

Order	Location	Default

1	/stats/configpath setting in rendermn.ini	.:${RMANTREE}/etc
2	`RMAN_STATS_CONFIG_PATH` environment variable override	none
3	prman -statsconfig </path/to/filename.ini>	stats.ini

If you specify an absolute path on the command line it will override any requested search paths. This is a convenient way to do quick testing without having to modify an existing config file. For example, suppose you normally run with a certain configuration of listeners, but then want to do a render with details printed to the console about a specific metric or group of metrics. You could do a debugging run that temporarily overrides the default configuration in one of two ways - either by setting/pre-pending the environment variable override:

```
setenv RMAN_STATS_CONFIG_PATH /my/test/directory
prman -statsconfig debug_stats.ini complicatedScene.rib
```

Or by providing a full-path, absolute file name:

```
prman -statsconfig /my/test/directory/debug_stats.ini complicatedScene.rib
```

Both of these methods will load the stats configuration from "`/my/test/directory/debug_stats.ini`"

If a configuration file name is not specified via the `prman` command-line option `-statsconfig` then the default configuration filename "`stats.ini`" will be used.

Currently, configuration files are not merged if more than one is found. The last one found wins. Future releases will include advanced configuration mechanisms, including the merging of configuration files.

Disabling Live Stats for Farm Renders

The ability to attach to a render running on a farm is a feature planned for the future but not currently supported. The live stats system will spawn a background process `sbrokerd` which can interfere with process lifetime and cgroup management of farm processes therefore it is recommended that live stats be explicitly disabled for farm renders using a bespoke pipeline configuration file.

In a nutshell, an explicit configuration file is needed that sets the config option `liveStatsEnabled` to `0`.

The following example is the minimum configuration file needed to turn live stats off. Place the config file in a known location and either add the `prman -statsconfig` command-line option or set the `RMAN_STATS_CONFIG_PATH` accordingly.

Farm, live stats off: stats.ini

```
# stats.ini
# Roz Stats farm configuration file
[Session]
name "Farm Stats Session"

# Disable the internal live stats server
liveStatsEnabled 0
```

It is not recommended to disable this option in the default $RMANTREE/etc/stats.ini file as that will disable live stats for all renders, including DCC interactive renders. Please pass RenderMan the new `stats.ini` configuration file via one of the other methods described above.

DCC Configuration

A live stats configuration UI pane is available in all RenderMan bridge products, with the exception of Solaris which does not yet have support for the new stats system. In addition, advanced configuration with an INI file is also available through the use of the config environment variable. See below for DCC-specific details.

Blender

RfB uses the `prman` command-line mechanism as described above, including the use of the `RMAN_STATS_CONFIG_PATH` override environment variable.

Houdini

If the `RMAN_STATS_CONFIG_PATH` environment variable is set RfH will use that search path to look for a file named `stats.ini`.

If no file is found, or if that environment variable is not set then the default configuration will be used.

Katana

If the `RMAN_STATS_CONFIG_PATH` environment variable is set RfK will use that search path to look for a file named `stats.ini`.

If no file is found, or if that environment variable is not set then the default configuration will be used.

Additional configuration is also available in RfK through the following attributes:

> `prmanGlobalStatements.stats.configPath` (default: "`.`")
>
> `prmanGlobalStatements.stats.configFile` (default: "`stats.ini`")

These attributes are currently not exposed in PrmanGlobalStatements, they must be set via AttributeSet or OpScript at the moment. Below is an OpScript which exposes these two attributes as user args with defaults as listed above. Copy and paste into your Katana scene and modify the path and config file name as needed.

Configuration OpScript

```
<katana release="4.0v2" version="4.0.2.000001">
  <node name="__SAVE_exportedNodes" type="Group">
    <node baseType="OpScript" edited="true" name="AdvancedStatsConfiguration" ns_colorb="0.050000" ns_colorg="
0.260000" ns_colorr="0.090000" ns_errorGlow="0.000000" ns_fromContext="legacy" selected="true" type="OpScript"
x="287.350171" y="-212.346316">
      <port name="i0" source="GafferThree.out" type="in"/>
      <port name="out" type="out"/>
      <group_parameter name="AdvancedStatsConfiguration">
        <string_parameter name="CEL" value="((/root))"/>
        <string_parameter name="location" value="/root/world/location"/>
        <group_parameter name="script">
          <string_parameter name="lua" value="&#0010;-- Default: &apos;stats.ini&apos;&#0010;local configFile =
Interface.GetOpArg(&apos;user.configFile&apos;):getValue()&#0010;&#0010;-- Default: &apos;.:${RMANTREE}
/etc&apos;&#0010;-- Can be overriden with RMAN_STATS_CONFIG_PATH&#0010;local configPath = Interface.GetOpArg
(&apos;user.configPath&apos;):getValue()&#0010;&#0010;Interface.SetAttr(&#0010;    &apos;prmanGlobalStatements.
stats.configFile&apos;,     &#0010;     StringAttribute(configFile))&#0010;&#0010;Interface.SetAttr(&#0010;
&apos;prmanGlobalStatements.stats.configPath&apos;, &#0010;     StringAttribute(configPath))&#0010;&#0010;"/>
        </group_parameter>
        <string_parameter name="executionMode" value="immediate"/>
        <string_parameter name="applyWhere" value="at locations matching CEL"/>
        <string_parameter name="applyWhen" value="during op resolve"/>
        <string_parameter name="modifierNameMode" value="node name"/>
        <string_parameter name="modifierName" value="modifier"/>
        <string_parameter name="resolveIds" value=""/>
        <number_parameter name="recursiveEnable" value="0"/>
        <string_parameter name="disableAt" value=""/>
        <string_parameter name="inputBehavior" value="by index"/>
        <number_parameter name="multisampleUserOpArgs" value="0"/>
        <group_parameter hints="{}" name="user">
          <string_parameter hints="{&apos;widget&apos;: &apos;fileInput&apos;}" name="configFile" value="stats.
ini"/>
          <string_parameter expression="&apos;.:&apos;+getenv(&quot;RMANTREE&quot;, &quot;.&quot;)+&apos;
/etc&apos;" hints="{}" name="configPath"/>
        </group_parameter>
      </group_parameter>
    </node>
  </node>
</katana>
```

Maya

If the `RMAN_STATS_CONFIG_PATH` environment variable is set RfM will use that search path to look for a file named `stats.ini`.

If no file is found, or if that environment variable is not set then the default configuration will be used.

Solaris

If the `RMAN_STATS_CONFIG_PATH` environment variable is set RfH will use that search path to look for a file named `stats.ini`.

If no file is found, or if that environment variable is not set then the default configuration will be used.

Houdini/Solaris Known Issue: Live stats will only work properly with either Solaris or Classic Houdini. If a scene is being rendered in one and then switched to the other the live stats will no longer work reliably. If this happens the scene must be closed and reopened, or the application must be restarted.

Diagnostic Reports

Advanced users may also use the configuration file to build and configure a list of presentation Listeners, each with a set of rules for metric data to be observed by that listener.

Checkpoint Render Report

For example, below is an example of a configuration file that enables an end-of-render JSON report, including checkpoints:

Checkpoint, JSON report: stats.ini

```
# checkpoint_stats.ini
# Stats default configuration file
#
# Copy this file to: /a/path/of/your/choosing
# then run:
# prman -statsconfig /a/path/of/your/choosing/checkpoint_stats.ini

version 0.1

# Set this to 1 for extra information when parsing this file
verbose 1

# Stats processing log level.
# Range is 0 (none) to 5 (debug output). Default is 3 (warnings)
logLevel 5

# Session configuration
[Session]
name "Checkpoint Session"

# List of listeners which the session should create and manage.
[ManagedListeners]

# JSON output report listener
[Listener]
    type "jsonreport"
    name "jsonListener"
    outputFilename "checkpoint_stats.json"
        keepAllCheckpoints 1

[MetricRules]
    # Save all metrics to the JSON file, sampled once per second
    [Rule]
    regexp ".*"
    samplingInterval 1000
```

See JSON Report Listener reference page for JSON output configuration options.