Configuration of Stats

Overview

RenderMan diagnostics have a built-in configuration for live data that is always presented in the `it` tool, toggled on with the "V" (capital "vee") keyboard shortcut. For DCC plugins the live stats are also automatically available in a separate Live Stats panel.

The end-of-render JSON Report must be explicitly enabled in a similar manner to the legacy XML Report, by providing a path to the output filename in the render options which are available in all DCCs.

```
Option "statistics" "jsonFilename" [ "filename.json" ]
```

Advanced configuration of the JSON report and presentation Listeners is available through the use of an ini-style configuration file.

Configuration File

The stats configuration file (default name: "stats.ini") defines the settings for the stats session (i.e. the collection of stats in a single render). The ini file can also be used for advanced configuration to build a list of listeners to attach, and per-listener rules for metric data to be observed by each listener.

Most of the high-level settings can be left at their default values. Custom sessions should be given their own names, and the stats component's own verbosity level can be adjusted for troubleshooting issues with Listeners or other stats processing.

Any number of listeners can be added in the Managed Listeners section, following the examples below and in the individual Listeners documentation.

File Format

The configuration file is broken out into sections. Currently it supports only a single Session with any number of Listener blocks. Each Listener block can have any number of Metric Rules. Lines that start with a '#' character are ignored.

Expand the code block below for an example of a configuration file that enables an end-of-render JSON report, including checkpoints.

checkpoint_stats.ini

```
# checkpoint_stats.ini
# Stats default configuration file

# General Options
version 0.1
verbose 0
logLevel 3

# Session Options
[Session]
name "Checkpoint Session"
liveStatsEnabled 1

# 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
```

General Options

These appear at the beginning of the file, before the Session Properties.

Option	Meaning	Type	Default	Required?
`version`	File format version. Allows backward compatibility should we change the syntax of session config files.	float	0.1	No
`verbose`	Parser output verbosity level. 0 = quiet, 1 = verbose.	int	0	No
`logLevel`	Sets the verbosity of the stats system. The available values range from 0-5: None, Error, Warn, Info, Debug	int	3 (Warn)	No

Session Options

The following table shows the options which can be set for a stats session (i.e. the diagnostics for a single render).

Session options are specified immediately following the `[Session]` section heading in the configuration file.

Option	Meaning	Type	Default	Required?
`name`	The name of the session. It's recommended to set this to a unique string when doing advanced configuration.	string	"Default Session"	Yes
`liveStatsEnabled`	Enables or disables the internal live stats server.	int (0/1)	1	No

Listener Options

The `[ManagedListeners]` section can contain one or more `[Listener]` blocks. All Listeners have a common set of required options, and each listener will have unique options to control its behavior. Some Listeners require no options.

Listener options are specified immediately following the `[Listener]` section heading. Each listener must have a type and a name. Any options added after those will be interpreted by the listener when it starts up. Unrecognized options will be ignored, while missing options will be defaulted to safe values.

Property	Meaning	Type	Required?
`type`	The type of Listener. Will be matched to a built-in listener type from the known list. If unmatched, will be assumed to be a plugin .so filename.	string	yes
`name`	The name of this Listener. Must be unique.	string	yes

Metric Rules

The `[MetricRules]` section follows the Listener options and can contain one or more `[Rule]` blocks. Each rule block must contain a `regex` property indicating which metrics that listener should be observing. Sampling options may be adjusted as well, as follows.

Property	Meaning	Type	Default	Required?
`regexp`	Specifies a regular expression which will be used to match metric names for this rule.	String		yes
`samplingInterval`	Specifies the number of milliseconds between samples of metrics which match this rule.	int	1000ms (once per sec)	no
`sampleImmediately`	Specifies whether metrics which match this rule should be sampled immediately.	int	0	no
`reportingRate`	For event metrics, specifies that a limit of only every N events of this type should be reported.	int	0 (don't limit)	no

Search Order

When rendering a RIB with the prman executable, the system will search for a stats configuration file named “stats.ini” in the search path defined by the " /stats/configpath" setting in rendermn.ini. The search path can be overridden using the RMAN_STATS_CONFIG_PATH environment variable, and the -statsconfig command line option to prman allows a further override of both the path and the filename.

Ways to specify configuration file location (RIB renders) in **order of precedence**:

Order	Method	Setting	Content	Default
1	Override “stats.ini” filename	prman -statsconfig <path+filename>	Filename with optional path (absolute or relative).	stats.ini
2	Override the rendermn.ini default search path	`RMAN_STATS_CONFIG_PATH`	Colon separated search path.	none
3	Default search path from rendermn.ini	/stats/configpath	Colon-separated search path.	`.:${RMANTREE}/etc`

- If prman is not provided a config file name via the *-statsconfig* command-line option then the default configuration filename "stats.ini" will be used.
- If no path is given to the *-statsconfig* command-line option then the search path defined by the environment variable is used, or the default in rendermn.ini if the environment variable is not set.

Examples

Using the example configuration file from the **File Format** section above, the command line might include:

```
prman -statsconfig /path/to/config/file/checkpoint_stats.ini /path/to/rib/scene.rib
```

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”.

DCCs will not pick up the default from rendermn.ini and generally cannot override the stats config filename and so must rely on the environment variable for configuration file search path control. See DCC Configuration below for details.

Filename expansion

Listener settings that specify a filename can include environment variables which will be expanded when processed by the Listener. Standard environment variable formats are supported, as well as Linux shell variable expansion syntax:

```
$VAR
${VAR}
${VAR:-fallback}
${VAR:=fallback}
```

For example:

```
outputFilename "${HOME}/checkpoint_stats.json"
```

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

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.

Listener control and configuration are not yet dynamic. In most cases, a render must be restarted in order to see the configuration change. In the case of RenderMan for Maya and Blender, the DCC application will need to be restarted.

DCC Configuration

A live stats configuration UI pane is available in all RenderMan bridge products. In addition, advanced configuration with the “stats.ini” configuration file is also available through the use of the `RMAN_STATS_CONFIG_PATH` 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. Interactive configuration of live stats is available in the Blender preferences.

Katana

RenderMan for Katana will search for a configuration file with the following precedence:

1. `The prmanGlobalStatements.stats` options take precedence over all other options if they are configured (see OpScript below).
2. `$RMAN_STATS_CONFIG_PATH/stats.ini` if the environment variable is set.
3. `$CWD/stats.ini`
4. `$RMANTREE/etc/stats.ini`

Advanced configuration is available in RfK through the following attributes:

`prmanGlobalStatements.stats.configPath` (default: “`.:$RMANTREE/etc`")

`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.

```
<katana release="6.5v1.010030b" version="6.0.1.000003">
  <node name="__SAVE_exportedNodes" type="Group">
    <node baseType="OpScript" edited="true" name="AdvancedStatsConfiguration" ns_colorb="0.05" ns_colorg="0.26"
ns_colorr="0.09" ns_errorGlow="0.0" ns_fromContext="legacy" selected="true" type="OpScript" x="493.466" y="
-240.278">
      <port name="i0" source="PrmanGlobalStatements.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=" -- 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;Interface.SetAttr(&apos;prmanGlobalStatements.stats.configFile&apos;,
StringAttribute(configFile)) &#0010;Interface.SetAttr(&apos;prmanGlobalStatements.stats.configPath&apos;,
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="
telemetry_stats.ini"/>
          <string_parameter expression="&apos;.:&apos;+getenv(&apos;HOME&apos;, &apos;.&apos;)+&apos;/stats
/configs&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.

Houdini/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.