


JSON Report

JSON file output, checkpoint/resume support

Description

Metric data gathered is written to a hierarchical JSON file. A final report is written at render termination, with intermediary stats written on checkpoint-exit events. Optionally (default: True) the data from each checkpoint will also be written.

 The end-of-render report requires knowledge of a render's endpoint in order to write out the final report. With live renders (e.g. Solaris), there really is no "end" to a render so the report isn't written out until the render is terminated.

Configuration

Required type for configuring the JSON report listener from an `ini` file.

`type "jsonreport"`

Options

Listener Options	Type	Description	Required?	Default
type	string	Must be "jsonreport"	y	
name	string	Unique name of the listener	y	""
outputFilename	string	File name for the final JSON report	y	""
keepAllCheckpoints	bool	Turn this off to save space by only writing out JSON stats immediately before exit, otherwise, a block of stats is written on every checkpoint during the course of the render.	n	y
Metric Rules				
regex	string	One or more metrics to be observed. If this is empty or missing no metric data will be written to the JSON file.	n	""
samplingInterval	integer	Sampling interval in MS for the metrics requested in the preceding <code>regex</code> param.	n	1000ms

The `outputFilename` setting 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}
```

Metric Data

The JSON report defines a "metrics" block which contains data payloads for all observed metrics, hierarchically organized according to each metric's namespace.

The format for an individual metric directly reflects the namespace built into the metric name, with each component in the namespace represented by a single, nested JSON object. For example, an entry for the metric:

```
/rman/raytracing.numRays
```

Would be the following:

```
{
  "metrics": {
    "rman": {
      "raytracing": {
        "numRays": {
          "description": "Total number of rays traced.",
          "timestamp": 2429320,
          "payload": [ 5873490 ]
        }
      }
    }
  }
}
```

Other metrics in the "raytracing" metric namespace would be nested under the "rman" block.

Analysis Tool

The viewing of the diagnostic report is best done through the new RenderMan [Stats Portal](#) application which can be used for both live data presentation and offline report introspection. Its content is currently limited, but useful for viewing the JSON metric hierarchy and comparing two or more reports.

Configuration Example: All metrics

The following excerpt from a configuration file will enable the JSON report output, writing to: `checkpoint_stats.json`. The "MetricRules" section indicates that all metrics should be gathered, with data sampled once per second. See below for an example output for this configuration.

checkpoint_stats.ini

```
# Roz Stats default configuration file
# Copy this file to: /a/path/of/your/choosing
# set RMAN_STATS_CONFIG_PATH = /a/path/of/your/choosing
# or run prman -statsconfig thisfile.ini
version 0.1

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

# Session configuration
[Session]
name "JSON Report Session"
liveStatsEnabled 1

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

Output Example

```
prman -statsconfig checkpoint_stats.ini -recover 1 scene.rib
```

The full metric data block has been omitted for brevity, however the header contains example metrics showing the metric description, time stamp at which the last value of the metric was reported, and the metric data payload.

checkpoint_stats.json

```
{
  "header": {
    "version": "26.0",
    "variant": {
      "description": "Riley render variant.",
      "timestamp": 178116,
      "payload": [
        "ris"
      ]
    },
    "maxsamples": {
      "description": "The maximum number of samples used by the integrator.",
      "timestamp": 2429320,
      "payload": [
        64
      ]
    },
    ...
  },
  "frame": {
    "attempts": [
      {
        "iteration": 20,
        "metrics": {...},
        "reason": "checkpoint"
      },
      {
        "iteration": 62,
        "metrics": {...},
        "reason": "checkpoint"
      },
      {
        "iteration": 83,
        "metrics": {...},
        "reason": "exiting"
      },
      {
        "metrics": {...},
      }
    ]
  }
}
```

Configuration Example: Memory tracking

The following excerpt from a configuration file will enable the JSON report output, writing to: `checkpoint_stats.json`. The "MetricRules" section indicates that all metrics should be gathered, with data sampled once per second. See below for an example output for this configuration.

json_memory_report.ini

```
# Roz Stats default configuration file
# Copy this file to: /a/path/of/your/choosing
# set RMAN_STATS_CONFIG_PATH = /a/path/of/your/choosing
# or run prman -statsconfig thisfile.ini
version 0.1

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

# Session configuration
[Session]
name "JSON Report Session"
liveStatsEnabled 1

# List of listeners which the session should create and manage.
[ManagedListeners]
[Listener]
    type "jsonreport"
    name "Memory Report"
    outputFilename "memory_report.json"
    keepAllCheckpoints 0

[MetricRules]
[Rule]
    regexp "/system/memoryTracking.*"
```

Output Example

```
prman -statsconfig json_memory_report.ini -variant xpu scene.rib
```

The bulk of the data is omitted for brevity however the report below shows an example of the memory breakdown of an XPU render.

Each memory metric is comprised of an empty-string description, the time stamp at which the latest memory update was received, and the metric payload. Every memory tracking payload contains three floats representing the amount of memory in bytes as described in the table below:

Payload Index	Description
0	The current process memory, including allocations which have not been committed to memory (current, or virtual).
1	Allocations which have been committed to memory (resident).
2	The maximum level of allocations committed to memory so far since the process began (max).

checkpoint_stats.json

```
{
  "header": {
    "version": "26.0"
  },
  "frame": {
    "attempts": [
      {
        "metrics": {
          "system": {
            "memoryTracking": {
              "system": {
                "rman": {
                  "geometry": {
                    "mem": {
                      "description": "",
                      "timestamp": 7769735,
                      "payload": [
                        0,
                        123279324,
                        0
                      ]
                    }
                  },
                  "volumes": {
                    "mem": {
                      "description": "",
                      "timestamp": 7769735,
                      "payload": [
                        0,
                        113004768,
                        0
                      ]
                    }
                  }
                }
              }
            }
          },
          ...
        },
        "gpu0": {
          "rman": {
            "scene": {
              "geometry": {...}
            },
            ...
          }
        }
      }
    ]
  }
}
```