Reading Activity.log

Activity.log can be helpful as it shows information on the time users logged in, the reports requested by users and the reports that are successfully generated.

The following is the guide is for **v7.5.0 and below**:

These are the messages that can be seen in activity.log:

2009-03-07 09:41:38,262,INFO ,admin , login from 127.0.0.1
Explanation: User logged into the server. In this case, 'admin' has logged into the server.

2009-03-07 11:29:14,756,INFO ,admin , render "/ElixirSamples/Report/Charting/3D Visualization.rml"
Explanation: It Is added when the user submits the request to the report renderer. This message will also appear if the report is on the queuing list. In this case, 'admin' has requested a report '/ElixirSamples/Report/Charting/3D Visualization.rml'

2009-03-07 11:29:14,762,INFO ,admin , Task added: Report 0 /ElixirSamples/Report/Charting/3D Visualization.rml
Explanation: The report renderer registers the rendering with the task manager (http://localhost:8080/tool/admin/tasks.html). This is the part when the report started to generate.

2009-03-07 11:29:29,763,INFO , , Task Timeout: Report 0 /ElixirSamples/Report/Charting/3D Visualization.rml (after 15) seconds
Explanation: The task has timed out.

2009-03-07 11:29:32,836,INFO ,admin , Task finished: Report 0 /ElixirSamples/Report/Charting/3D Visualization.rml

Explanation:

When the report has rendered, the activity is de-registered from the task manager (http://localhost:8080/tool/admin/tasks.html) which also mean the task has finished and is no longer consuming CPU or memory.

Although the task is finished, it does not indicate whether it was successful or not generated. The task manager just records that a task is being worked on, and then records when the task is no longer being worked on (for whatever reason).

2009-03-07 11:29:32,845,INFO ,admin , render "/ElixirSamples/Report/Charting/3D Visualization.rml",elapsedTime: 18071 pageCount: 2 byteCount: 262573 mimeType: application/pdf statusCode: 1

Explanation:
It means the data is all processed for reply and any failure from this point onwards is a network failure - nothing to do with data or the report, as typically the user has closed the connection.

The statistics that are output describe the rendering activity and show the summary of the report generated.

Note:
This also means that the report has been written to the output. The output might be buffered and not yet read by the client. Indeed the client may have disconnected without waiting for the reply to complete. We can confirm we have sent the response, but that does not guarantee that the user has successfully received the output.

Example:
If a user closes the connection while the report is being rendered, or while it is being streamed back to the browser, then the user may not get this report summary because it has not been completely written.

2009-03-07 11:39:32,812,INFO , , Task removed: Report 0 /ElixirSamples/Report/Charting/3D Visualization.rml
Explanation: Report has been removed from the task manager (http://localhost:8080/tool/admin/tasks.html). From v7.2, you can control the task manager property from ERS2.xml: <!-- Task Manager (experimental) --> <ers:mbean name="ERS2:name=ReportMonitorThread" <ers:property name="Timeout">0</ers:property> <ers:property name="MonitorInterval">60</ers:property> <ers:property name="FlushInterval">600</ers:property> </ers:mbean> Report Monitor Thread Settings - Timeout (secs) : How long to wait before cancelling the report - Monitor Interval (secs) : How often to check if a report has exceeded the timeout time - Flush Interval (secs): How long to keep the tasks in the task list once they have completed/timed out. Why do we need to kill the thread in the task manager? If the report goes into an endless loop, eg. a JavaScript script says for (var i=0;i<10;j++) which will loop forever because j changes, but i is always <10. Then the report monitor can detect this and kill the task, otherwise it will consume resources and CPU until the server stops.

The following is the guide for **v7.5.1 and above**.

These are the messages that can be seen in the activity log:

2009-04-11 16:34:30,453,INFO ,admin , login from 127.0.0.1
Explanation: User logged into the server. In this case, 'admin' has logged into the server.

2009-04-11 16:36:32,281,INFO ,admin , render "/ElixirSamples/Report/Charting/3D Visualization.rml"

Explanation:
It Is added when the user submits the request to the report renderer. This message will also appear if the report is on the queuing list.
In this case, 'admin' has requested a report: '/ElixirSamples/Report/Charting/3D Visualization.rml'

2009-04-11 16:36:32,390,INFO ,admin , Task added: Report 0 /ElixirSamples/Report/Charting/3D Visualization.rml

Explanation:
The report renderer registers the rendering with the task manager (http://localhost:8080/tool/admin/tasks.html). The status appear as 'Started'. This is the part when the report started to generate.

2009-04-11 16:38:40,625,INFO ,admin , Task completed: Report 0 /ElixirSamples/Report/Charting/3D Visualization.rml

Explanation:
When the report has rendered, the activity is de-registered from the task manager (http://localhost:8080/tool/admin/tasks.html).
The status appears as 'Done'. For more information, look at the final message from the module that processed the task.

2009-04-11 16:24:01,875,INFO ,admin , Task failed: Report 0 /ElixirSamples/Report/Charting/3D Visualization.rml

Explanation:
When the report has rendered, the activity is de-registered from the task manager (http://localhost:8080/tool/admin/tasks.html).
The status will still appear as 'Started'. This is the part when the report has failed to generate. (Need to refer to server.log for the full error message).

2009-04-11 16:38:40,625,INFO ,admin , render Report 0 "/ElixirSamples/Report/Charting/3D Visualization.rml",elapsedTime: 18071 pageCount: 2 byteCount: 262573 mimeType: application/pdf statusCode: 1

Explanation:
The statistics are output that describe the rendering activity. This shows the summary of the report generated*.

Note:
This also means that the report has been written to the output. The output might be buffered and not yet read by the client. Indeed the client may have disconnected without waiting for the reply to complete. We can confirm we have sent it, but that does not guarantee that the user has successfully received the output.

Example:
If you close the connection while the report is being rendered, or while it is being streamed back to the browser, then you may not get this report summary because it has not been completely written.

*If the Task failed (2009-04-11 16:24:01,875,INFO ,admin , Task failed: Report 0 /ElixirSamples/Report/Charting/3D Visualization.rml), this summary message will not display as it is not written to the output.

The following are the Status Codes:
statusCode -1 = NOVALUE
statusCode 0 = STATUS_UNKNOWN
statusCode 1 = STATUS_OK
statusCode 2 = STATUS_NO_DETAILS
statusCode 3 = STATUS_NO_DATASOURCE
statusCode 4 = STATUS_LOGGED_EXCEPTION
statusCode 5 = STATUS_CANCELLED
statusCode 6 = STATUS_TIMEOUT
statusCode 7 = STATUS_NO_REPORT
statusCode 8 = STATUS_UNABLE_TO_COMPLETE
statusCode 9 = STATUS_NO_JOB

2009-04-11 16:48:40,625,INFO , , Task removed: Report 0 /ElixirSamples/Report/Charting/3D Visualization.rml

Explanation:
Report has been removed from the task manager (http://localhost:8080/tool/admin/tasks.html).
From v7.2, you can control the task manager property from ERS2.xml:

<!-- Task Manager (experimental) --> <ers:mbean name="ERS2:name=ReportMonitorThread" <ers:property name="Timeout">0</ers:property> <ers:property name="MonitorInterval">60</ers:property> <ers:property name="FlushInterval">600</ers:property> </ers:mbean>

Report Monitor Thread Settings:
- Timeout (secs) : How long to wait before cancelling the report
- Monitor Interval (secs) : How often to check if a report has exceeded the timeout time
- Flush Interval (secs): How long to keep the tasks in the task list once they have completed/timed out.

Why do we need to kill the thread in the task manager?
If the report goes into an endless loop, eg. a JavaScript script says for (var i=0;i<10;j++) which will loop forever because j changes, but i is always <10. Then the report monitor can detect this and kill the task, otherwise it will consume resources and CPU until the server stops.