GC Analyzer

User Manual

Contents

[Time Origin 3](#_Toc365655458)

[gc-analyzer Development 4](#_Toc365655459)

[TODO 4](#_Toc365655460)

# Time Origin

The GC logs do not maintain a full time stamp, but just an offset, expressed in milliseconds, from the moment the logging started. In order to calculate a timestamp for an event, a “time origin” is needed. gc-analyzer accepts a time origin value specified on command line, with the following syntax:

-o|--time-origin MM/dd/yy HH:mm:ss,SSS

The hour is represented as a 0-23 integer. For example, to specify December 29, 2011 8:08:08 PM use 12/29/11 20:08:08,000

If a time origin is not specified, gc-analyzer will try to infer it from the name of the file, if possible, or otherwise will throw an exception, warning the user that a time origin is needed. Example of supported patterns:

\*.29-Dec-11-0808 (time origin will be initialized to 12/29/11 8:08:00 AM)

\*.29-Dec-11-220344 (time origin will be initialized to 12/29/11 10:03:44 PM)

If the time origin can be inferred from the name of the file (the name of the file is among the known patterns) and a time origin is specified on command line, the command line value takes precedence.

# gc-analyzer Development

## TODO

* **gca 04/28/15.** Next on gca - I should be able to export all fields by default. If a field does not apply, it should generate ‘’ in the CSV file.
* **gca 04/28/15.** I should not throw any exception and not stop parsing, just log a warning and go on. This way I can extract **some** info from a file instead of stopping.
* **gca 04/28/15.**  standardize the automatic maven dependency pull from pom.xml into the final artifact (release.sh)
* Add garbage\_collection.log-08-28-2013\_06-40-47 to src\test\resources\collected and write a test that passes around it.
* Clarify ParserException vs Exception in GCEventParser.parse(..)
* What is the difference between the first and second duration in the example. When figuring out go to TODO\_swev

1.985: [GC 1.984: [ParNew: 136320K->6357K(153344K), 0.0083580 secs] 136320K->6357K(4177280K), 0.0085020 secs] [Times: user=0.05 sys=0.01, real=0.01 secs]

* Implement expression: heap-after - og-after
* Should be able to calculate time from previous collection (and time from previous collection of the same type)
* Define strategy on log.info() vs System.out.print. In a command line environment I need the utility to be "quiet" - not generate any undesired output.
* Re-implement it in such a way to make sure that after a pass, I "understand" every bit of that gc log file, and if there are bits I don't understand, I throw an exception. This should be the default behavior. Then, I should have the option to turn the --strict behavior on, and extract as much as possible, with warnings as comments embedded in the output file.
* The gc log analysis code should be designed in such a way that it runs from a command line interface and from a server process.
* Read about parallel GC - update wiki.
* Currently I am building the distributable ZIP in a separated build.xml file - integrate this with Maven "the right way" - one module that builds the JAR and one module that builds the ZIP.
* Remove all remnant gc analysis classes classes from Universus, keep Universus clean from gc analysis.