## Lumberjack – A Logger with Class

Lumberjack is a class which implements general purpose logging utility. It provides you with a tool to log almost any type of activity. A log file is created for each day. The log file names are in the format **YYYYMMDD.log**, optionally prefixed by a character vector of your choosing.

When started, Lumberjack will spawn a new thread and maintain a cache of log records to be written to file. Your application will call the Lumberjack's Log method to append log records to the cache. You may specify how often Lumberjack should write the cache to file.

Lumberjack has four public fields/properties:

Directory – the directory to contain the log files

Prefix – prefix for the log file names

Interval – the interval, in seconds, to wait between writing the cache to file

Active – a Boolean flag indicating the whether Lumberjack is running

Lumberjack has three public methods:

{*flags*} Log *record* – Write a log record to the cache

*flags* is an optional argument of up to 2 Boolean elements indicating

[1] Prepend a timestamp in the form YYYY-MM-DD@HH:MM:SS.sss to

the record. The default is 0.

[2] Append an appropriate End Of Line (EOL) sequence to the record.

The default is 0.

Start – starts Lumberjack

Stop – stops Lumberjack

To initialize Lumberjack, you can either call its

* niladic constructor and then assign values to its fields

pb←⎕NEW Lumberjack

pb.Directory←'c:\mylogs'

pb.Interval←15

pb.Prefix←'myApp'

pb.Active←1

**Note:** because setting Active to 1 will start Lumberjack, this should be the last value you set

monadic constructor with values for the fields directly  
 pb←⎕NEW Lumberjack (Directory Interval Prefix Active)

Example: pb←⎕NEW Lumberjack ('c:\mylogs' 15 'myApp' 1)

Assuming the date is 1 October 2012, either option above would create a file named:

**c:\mylogs\myApp20121001.log**

To start Lumberjack, you can:

* Pass a value of 1 for Active when calling its monadic constructor
* Call the Lumberjack.Start method
* Assign a value of 1 to the Active property

To stop Lumberjack, you can:

* Call the Lumberjack.Stop method
* Assign a value of 0 to the Active property

To create log records, use the Log function

1 1 pb.Log 'Start Transaction'

⍝ run some application code

1 1 pb.Log 'End Transaction'

The resulting log file will look like:

2012-10-11@18:11:06.045 Start Transaction

2012-10-11@18:11:10.786 End Transaction

Lumberjack uses the Files utility namespace.

## Building Your Own Logger Based on Lumberjack

Lumberjack will write whatever data you passed to its Log method. As such, you may want to build your own logger derived from Lumberjack.

You may want to initialize your logger based on some configuration file settings, and produce log records in a specific format.

The example below is from MiServer, Dyalog's APL-based web server. It reads its configuration settings from an XML file. It writes log records for every HTTP Request received by MiServer. The records are formatted according to the Common Logfile Format used by many web servers[[1]](#footnote-1) by formatting the records and are written by calling ⎕BASE.Log.

The code below has been formatted/wrapped to fit within the margins of this page.

:Class Logger : Lumberjack ⍝ MiServer HTTP Request Logger based on Lumberjack

⍝∇:require =/Lumberjack

⎕ML←1

missing←{0∊⍴⍵:'-' ⋄ ⍵}

∇ Make ms;file;log;directory

:Access public

:Implements Constructor :Base

⍝ Check that configuration file exists

('MiServer Logger configuration file "',file,'" not found')⎕SIGNAL

11/⍨~#.Files.Exists file←ms.Root,'Config/Logger.xml'

⍝ parse the configuration file and assign logger fields

:If ~0∊⍴log←(#.XML.ToNS #.Files.GetText file).Logger

Prefix←log #.Boot.Setting'prefix' 0 ''

Interval←log #.Boot.Setting'interval' 1 10

directory←#.Boot.SubstPath log #.Boot.Setting'directory' 0 ''

Directory←{⍵,(~'/\'∊⍨¯1↑⍵)/'/'}directory

Active←log #.Boot.Setting'active' 1 0

:Else

('Invalid MiServer Logger configuration file "',file,'"')⎕SIGNAL 11

:EndIf

∇

∇ Log req ⍝ write Common Logfile Format record

:Access public

⎕BASE.Log((missing 2⊃req.PeerAddr),' ',  
 (missing req.Session.User),#.Dates.LogFmtNow,'"',  
 req.Command,' ',req.Page,'"',  
 ∊' '∘,∘⍕¨req.(Response.Status MSec Response.Bytes)),EOL

∇

:EndClass

A sample MiServer Logger Record

127.0.0.1:49330 - [08/Jun/2012:22:23:58 +0000] "get /index.dyalog" 200 164 3879 1638

Each MiServer Logger log record consists of the following 8 fields. If field does not have a value, it is replaced with a dash:

1. <IP address>:<Port> 127.0.0.1:49330
2. <User ID> -
3. [<Timestamp in UTC>] [08/Jun/2012:22:23:58 +0000]
4. "<HTTP command> <Resource>" "get /index.dyalog"
5. <HTTP status code> 200
6. <Millseconds to send response>` 164
7. <Size of response before compression> 3879
8. <Size of response after compression> 1638

1. <http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format> [↑](#footnote-ref-1)