**Exchequer Logging Sub-System (Version 1.0)**

**Overview**

The Exchequer Logging Sub-System is designed to enable a simple but expandable logging functionality to be added into applications. The main criteria used are:

* *Logging should impact as little as possible on performance.* To achieve this, all writing to disk is done in a low priority thread.
* *Any errors in logging should not affect the application.* Consequently, the system is designed to handle exceptions quietly and to terminate if logging is not possible for any reason.
* *Logging should be configurable to allow different levels of logging to be configured for different areas of the application without changing the application.* The logging levels for each area are held in an ini file and allow the following levels:

Logging off (0)

Log errors (1)

Log information and errors (2)

**Components of the logging sub-system.**

The sub-system consists of four separate components. (All units are in Entrprse\Funcs.)

***The configuration object.*** (TEntLogIniFile in EntLogIniClass.pas).

This object reads an .ini file (currently EntLog.ini) from the data directory of the application. To avoid performance issues, the file is only read once, when logging starts. The ini file contains the name of the log file to write (always in the logs directory of the data directory) and the logging level for each area.

The user does not need to access this class directly.

***The log queue.*** (TEntLogQueue in EntLogQueueClass.pas).

This singleton object is a queue which holds log lines until they are written to disk. The object is accessed via a critical section to avoid threading conflicts.

The user does not need to access this class directly.

***The log writer.*** (TEntLogWriter in EntLogWriterClass.pas).

This is a thread object with priority of tpLowest which runs continually. It removes the next line from the log queue and writes it to the log file. When the thread is terminated it will write any remaining lines from the log queue before closing. If the thread terminates because of a fatal IO error, then it will not try to write to the log file but will write a message to the Windows Event Log.

The user does not need to access this class directly.

***The logger.*** (TEntBaseLogger, etc. in EntLoggerClass.pas).

This is the class which the use needs to access to write to the log. The base class exposes two methods (LogError and LogInfo) which the user can call. The logger object will use the configuration object to decide whether the message passed in either method needs to be written to the log. If so, then it will prefix the method with a comma-separated set of standard information and add it to the log queue, returning immediately.

It is expected that a number of descendants of TEntBaseLogger will be written to make it simpler to add logging to specific areas of code. To begin with, TEntSQLReportLogger has been added with functions for logging SQL versions of standard reports. (StartReport, StartQuery, FinishQuery, FinishReport.)

**The .Ini files**

Each company in an Exchequer install will have its own ini file. If no ini file is found in the data directory for the company, then the file from the root company will be used. The ini file will be structured as follows:

[GLOBAL]

LogFileName=logfile.csv

[AREAS]

Default=1

SimplifiedTrialBalance=2

When an instance of TEntBaseLogger or any of its descendants is created, the constructor takes as a parameter a string which should specify the area of code which is being covered – e.g. for a report, it would probably be the report name. The logger will then check the values from the ini file (via the configuration object) looking for the string in the Areas section to determine the required level of logging. If the string is not found, then the Default level will be used. (If Default is not found, then logging will be turned off for that object.) The level of a logger is also exposed as a read/write public property so that it can be changed at run-time if necessary, overriding the value in the ini file.

**Output**

Each log line consists of the following data separated by commas:

Date, Time, Computer Name, Process ID, Thread ID, User ID, Application Name, Module Name (blank if not a DLL), Exchequer Version, Exchequer Directory, Name (Area), Level, Message, xd1, xd2, xd3

Xd1, xd2, and xd3 are fields for extra data. They are empty in the base logger but can be populated by descendants.

**Using the logging sub-system.**

Your application needs to start the logging sub-system before it can do any logging. (The sub-system must be explicitly started because it needs to know the data directory that the application is working with.) Logging is started by calling StartLogging (in EntLoggerClass.pas), passing in the required data path. This will automatically create the configuration object and load the values from the ini file. It will then create and start the log writer thread. After this you can create and use any number of logger objects. Logging is turned off automatically in the finalization section of EntLoggerClass.pas, although it can be turned off explicitly by calling StopLogging. (This would be needed in Enter1 when switching to a new company.)