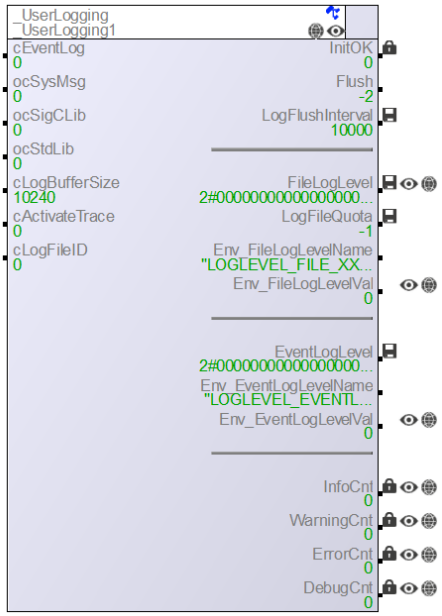
_UserLogging



This class allows the user to easily log Application Messages to:

1. Log Files (Using the _SysMsg Interface).
2. EventLog
3. LASAL Class 2 Debugger Trace.

When a log is created, the entry type, source and extra parameters are concatenated into a standard format string to allow for filtering the Applications log messages at will.

The user can also determine which levels of messages are logged and therewith control the log output on a per class instance basis.

Implementation

The _UserLogging class needs to be instantiated in the project by creating an instance of a derivative of this class (Such as _ComLogging), or by placing this class in a network. The user must then configure the instance for operation, to do so the user must set his preferred values to the relative clients and servers, these values have been initialised to default settings to suit a standard application, the most important values to be configured are:

1. cLogBufferSize – This client denotes the amount of Application Heap that is allocated for buffering Log Messages and is initialised to 10kB , the user must ensure that the value specified is optimal for his application (Large enough to allow for buffering messages before writing to disk and not too large).

2. FileLogLevel and EventLogLevel – These servers denote the levels of log messages that will be logged, where FileLogLevel filters messages to be logged to file and EventLogLevel filters messages to be sent to the Application EventLog.

   The following bits are used in the above mentioned filters:

   2#00001 = Info
   2#00100 = Warning
   2#01000 = Error
   2#10000 = Debug

   The FileLogLevel and EventLogLevel values can also be set by setting Environment Variables in the Autexec.lsl for loading at boot time or by CLI command (using AddToServiceProvider() or RemoteCli) during Runtime.
   The Environment Variable names to use for loading the FileLogLevel and EventLogLevel values can be freely chosen by the user, the name to be used for the respective value is set on the servers: Env_FileLogLevelName and Env_EventLogLevelName.

Example Log Entry:

An example Log entry looks as follows:

EMAP77;_COMLOGGING1;Error;44;1800;Log Text Here;2222;7777;1111;4444

„EMAP77“: Category of the log. This value could for example denote a communication protocol or an application section name.
„COMLOGGING1“: Instance name of the object that generated the Log Message.
„Error“: Level of the log message created.
„44“: Message Group Value assigned to the log. This value could be used to reference the Message to documentation.
„1800“: Message Number Value assigned to the log. This value could be used to reference the Message to documentation.
Log Text Here: The log message that was passed to the logging class.
„2222“: Parameter 1
„7777“: Parameter 2
„1111“: Parameter 3
„4444“: Parameter 4

Global Methods

AddEntry	Method to add an entry to the logging class. This method is Threadsafe, and consumes all values immediately when called, therefor values passed to this method are allowed to be changed or destroyed directly after execution. => pCategory: ^CHAR: Pointer to the category name the message is assigned to. Must be set. => Level: UDINT: Value used to denote the Log Level, value is interpreted bit-wise and allows for multiple levels to be assigned to a log message. Must be set > 0 & < 30. => MsgGroup: DINT: Message Group Number assigned to the log. => MsgNbr: DINT: Message Number assigned to the log. => pMsg: ^CHAR: Pointer to the first character of the Log message string. This string will be truncated if the message exceeds the maximum length. => pPara1: ^DINT: Pointer to the first Parameter to be added to the log message, if NIL the parameter is logged as empty text. => pPara2: ^DINT: Pointer to the second Parameter to be added to the log message, if NIL the parameter is logged as empty text. => pPara3: ^DINT: Pointer to the third Parameter to be added to the log message, if NIL the parameter is logged as empty text. => pPara4: ^DINT: Pointer to the fourth Parameter to be added to the log message, if NIL the parameter is logged as empty text. <= Retcode: t_e_AddEntryRet: Return value of the Method: AE_Success: Message successfully logged. AE_InputNotOK: The Category String or Log level was not OK. AE_Truncated: Message log but partly truncated.
Initialise	Method to initialise the object, this method is automatically called during Init and must only be called by the user if one of the following values are changed: 1. cLogFileID (Read during method execution). 2. cLogBufferSize (Read during method execution). 3. One of the Environment Variable Names. <= Retcodet_e_UserLogInitRet: Return value after execution of the method. UIR_Success: Successfully Inittialised. UIR_LogFileIDNotOK: Value on cLogFileID was not set / incorrect. UIR_MemAllocFail: Failed to allocate the memory required for the _SysMsg logging interface. UIR_SysLogOpenFail: Failed to open the log file using the _SysMsg interface.

Interface

Clients

cEventLog	Command Channel to the ClassSvr of the EventQueue class. Datatype: DINT
ocSysMsg	Object Channel to the _SysMsg OS Interface. Must not be connected. Datatype: DINT
ocSigCLib	Object Channel to the SigCLib OS Interface. Must not be connected. Datatype:
ocStdLib	Object Channel to the _StdLib OS Interface. Must not be connected. Datatype: DINT
cLogBufferSize	Size of the memory to be allocated for the Log File buffer, value denoted in bytes, see the _SysMsg Interface documentation for specific details. Datatype: UDINT
cActivateTrace	If this client is set to 1 all the valid log messages passed to the Class will be output to the LASAL Class 2 Trace mechanism. Datatype: UDINT
cLogFileID	The Log File ID to be used is written to this client, valid values are > 0 & <= 9. Where the number determines the Log File to be used for logging: EVENT1X.LOG, 1 -> EVENT11.LOG 2 -> EVENT12.LOG, etc. Datatype: UDINT

Server

InitOK	Object channel can be used to call the initialise method, the InitOK value also shows the result of Initialisation (0 = Failed, 1 = OK). Unit: - / Data Type: DINT Range: 0/1 / Write Protected: TRUE Default Value: 0 / Retentive: FALSE
Flush	If this server is written to the Log Data is flushed, return values are 0 = Success, -2 = No SysMsg Log File Handle, Other = SysMsg LFlush error return code. Unit: - / Data Type: DINT Range: 0 / Error Value / Write Protected: FALSE Default Value: - / Retentive: FALSE
LogFlushInter-val	This server defines the interval (milliseconds) at which the Log File should be flushed, setting this value too low could shorten the lifetime of the flash media as the disk is written to with every flush call. Unit: - / Data Type: UDINT Range: - / Write Protected: FALSE Default Value: 10secs / Retentive: File
FileLogLevel	This server shows the current logging filter for the log to file mechanism of the class. Unit: - / Data Type: BDINT Range: - / Write Protected: FALSE Default Value: 2#1100 / Retentive: FILE
LogFileQuota	This server shows the maximum file size (FileQuota) of the log file to be created, if this value is exceeded the log file is backed up and a new file is created. Maximum

	amount of log files kept = 2. Unit - Data Type UDINT Range - Write Protected FALSE Default Value - Retentive FILE
Env_File_LogLevelName	This server shows the name of the File Log Level Environment Variable. If the variable is not found, the log level is not altered. Unit - Data Type UDINT Range - Write Protected FALSE Default Value - Retentive FALSE
Env_FileLogLevelVal	This server shows the value set to the environment variable named in the server, Env_FileLogLevelName, value can also be written. Unit - Data Type DINT Range - Write Protected FALSE Default Value - Retentive FALSE
EventLogLevel	This server shows the current logging filter for the log to EventLog mechanism of the class. Unit - Data Type BDINT Range - Write Protected FALSE Default Value 2#0 Retentive FILE
Env_EventLogLevelName	This server shows the name of the Event Log Level Environment Variable. If the variable is not found, the log level is not altered. Unit - Data Type UDINT Range - Write Protected FALSE Default Value - Retentive FALSE
Env_EventLogLevelVal	This server shows the value set to the environment variable named in the server, Env_EventLogLevelName, value can also be written. Unit - Data Type DINT Range - Write Protected FALSE Default Value - Retentive FALSE
InfoCnt	The number of logs with level “Info” is shown here, the value increases regardless of the filter setup. Unit - Data Type DINT Range - Write Protected TRUE Default Value - Retentive FALSE
WarningCnt	The number of logs with level “Warning” is shown here, the value increases regardless of the filter setup. Unit - Data Type DINT Range - Write Protected TRUE Default Value - Retentive FALSE
ErrorCnt	The number of logs with level “Error” is shown here, the value increases regardless of the filter setup. Unit - Data Type DINT Range - Write Protected TRUE Default Value - Retentive FALSE
DebugCnt	The number of logs with level “Debug” is shown here, the value increases regardless of the filter setup. Unit - Data Type DINT Range - Write Protected TRUE Default Value - Retentive FALSE