|  |
| --- |
| Macmillan usa |
| Platform-X Logging |
| Logging infrastructure for Platform-X |
|  |
| **Dave Newman** |
| **8/25/2011** |

|  |
| --- |
| Description of the logging infrastructure used in Platform-X. |

Contents

[Logging API 3](#_Toc302043171)

[ILogger 3](#_Toc302043172)

[LogMessage 3](#_Toc302043173)

[ITraceManager 3](#_Toc302043174)

[ITraceHandle 3](#_Toc302043175)

[Logging Implementations 4](#_Toc302043176)

[Enterprise Library Logging Block 4](#_Toc302043177)

[Logging Database Schema 5](#_Toc302043178)

Platform-X Logging

The Platform-X Learning Management System (PX-LMS) makes heavy use of logging to report errors, status, and debugging information. The logging facilities are presented to the application through a simple API, which is currently implemented using the Microsoft Enterprise Library. Due to the way the API is structured, the dependency on Enterprise Library can be substituted with an alternative framework like Log4Net.

# Logging API

The logging API used by Platform-X is part of the Bfw.Common namespace. There are three interfaces that are used to abstract the logging implementation from the API the application uses. They are ILogger, ITraceHandle, and ITraceManager. Full documentation for these interfaces can be found at <http://dev.worthpublishers.com/docs>.

## ILogger

The [ILogger](http://dev.worthpublishers.com/docs/d3/de3/interface_bfw_1_1_common_1_1_logging_1_1_i_logger.html) interface allows the application to log messages. How and where logged messages are stored depends on the implementation. The documentation linked above provides the full signature for the interface, but the primary method is Log(LogMessage message), which all other Log overloads eventually call.

ILogger's methods are typically used to log errors, application status, or simple debugging information. The overloads provided for the Log method should cover most common cases like logging an exception, or a simple string.

## LogMessage

The [LogMessage](http://dev.worthpublishers.com/docs/dd/de8/class_bfw_1_1_common_1_1_logging_1_1_log_message.html) class is used to represent any message that can be logged by the API.

## ITraceManager

The [ITraceManager](http://dev.worthpublishers.com/docs/d9/d5c/interface_bfw_1_1_common_1_1_logging_1_1_i_trace_manager.html) interface specifies how trace information can be logged from the application. Traces differ from logging in that they are designed for performance tuning as opposed to error or status logging. Traces may be nested and may be stored differently from log messages depending on the implementation used.

## ITraceHandle

The [ITraceHandle](http://dev.worthpublishers.com/docs/d9/d2f/interface_bfw_1_1_common_1_1_logging_1_1_i_trace_handle.html) interface exposes no methods and simply inherits from IDisposable. This handle interface is returned by the ITraceManager in order to allow an individual trace to be disposed of explicitly. Typical usage is to allow the ITraceHandle returned by ITraceManager.StartTrace to be disposed when it exists scope, for example in a using statement.

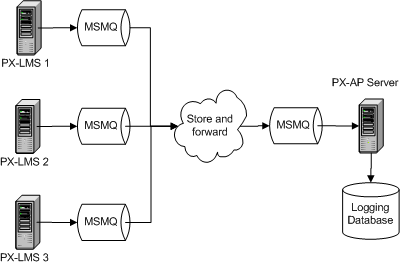
# Logging Implementations

As mentioned previously, the logging API abstracts the true implementation. Therefore, how and where log messages and traces are actually stored can vary greatly. This section covers the current implementation of the logging interfaces.

## Enterprise Library Logging Block

The currently implementation of the logging API is built using the Microsoft Enterprise Library Logging Application Block. This is a commonly used component that is freely available and open source (see [EntLib Documentation](http://entlib.codeplex.com/)).

Enterprise Library allows the application to configure different storage locations, formats, and filters to be applied to log messages. There are three common storage mediums used by PX-LMS to log messages and traces: System Event Log, Rolling Text File, and MSMQ/Database. The latter method is used in all environments other than local development, and can be expressed by the following diagram.



As can be seen in the above image, messages are logged to an MSMQ on each server that is configured to forward them to a central MSMQ. The MSMQDistributor service provided with Enterprise Library will then read the messages from the central queue and store them in the database.

This setup achieves two goals. First, log and trace messages can be written quickly with minimal impact on the PX-LMS application. Second, since all messages wind up in a central location the Platform-X Admin Portal (PX-AP) is able to aggregate information across all instances of the PX-LMS application.

## Logging Database Schema

The following image shows the default Enterprise Library logging database schema.

