DME3 Logging Summary

# Purpose

The purpose of this document is to identify and convey the specifics of logging functionality in the DME3 specification.

# Design

## Overview

The direction of the design has been specified to strictly use the SLF4J Logging Interface. It is further dictated that the implementation be independent of the services that the DME libraries provide, thus it will be entirely up to the client to provide a valid implementation that conforms to the SLF4J interface.

More information about SLF4J is available [here](http://www.slf4j.org/)

Several well-known and well-used implementations of SLF4J are [log4j](http://logging.apache.org/log4j/1.2/), [JCL (aka Apache Commons Logging)](http://commons.apache.org/proper/commons-logging/) and [Logback](http://logback.qos.ch/). More details about bridging legacy APIs are available [here](http://www.slf4j.org/legacy.html)

## Requirements

1. Design should rely on the SLF4J Interface
2. Design should not provide an implementation. This **must** be provided by the client.
3. Design should work with any implementation that conforms to the SLF4J implementation.
4. Logging messages that originate inside the DME library should include at a minimum:
   1. Date
   2. Log Level
   3. Hostname
   4. Process ID
   5. Thread ID
   6. Tracking ID / Conversation ID
   7. Service URI
   8. Class
   9. Method
   10. Original message

## API

The API provided by the DME libraries is a subset of those provided by the [SLF4J Logger](http://www.slf4j.org/api/org/slf4j/Logger.html). Because the provided API is implementation-independent, the presence of a configuration file is not an option, thus all parameters in item 4 above must be sent as parameters to the logging utility. In keeping with SLF4J’s design of optimized message construction, additional parameters which would be used to construct the original message may be sent in as well.

For instance, given the following:

log.debug( “This is the complex object we just saw: “ + reallyComplexObjectToSerialize.toString() );

It is much more efficient (especially when debug logging is not enabled) to write it as follows:

log.debug( “This is the complex object we just saw: {}”, reallyComplexObjectToSerialize);

This allows the expense of creating the full message to be delayed until after it can be verified that the log level is set appropriately.

## Classes

* DMELogInit – because not having an SLF4J implementation is a fatal error for anyone using DME, DMELogInit.init() must be called somewhere in the bootstrapping of the API. Normally SLF4J gracefully handles the lack of an implementation by printing a warning and defaulting to a “no-op” implementation.
* ConversationId – Container class for conversationId. Presently contains one member variable defined as a Long. It is up to the specific module using the logger to properly handle setting this value.
* LogUtil – Main utility class for logging internal DME messages. Contains multiple method signatures for logging all information required as well as convenience methods for passing in 4(a) (Date) as a Long as well as 4(i) (Method) as a String.

