

# Audit Trails

## Introduction

Transformation Manager (TM) provides two distinct means for producing an audit trail of transform processing - **Logging** and **Error Handling**. Logging produces text stream output, whereas Error Handling calls an error project to process errors that have been noted. They can also be combined where an error that has been noted can make a logging request in the `CATCH_ERROR` block.

## Logging

The two main types of logging supported are:

- **System Generated Logging**
- **Logging Functions**

### *System Generated Logging*

TM generates system messages as a matter of course, for example the activity times or full report of activities can be requested, where an activity is a stage of transformation processing. The extent of the system generated logging is controlled by setting a logging level and the default settings are set for normal usage.

The built-in system generated logging functions are:

`LOADUSERREFERENCEOBJECT` - loads a `UserReferenceObject`

`SETFULLREPORT` - sets the system to display a full report of activity on the console.

`SETLOGBUFFER` - sets the buffer size used by the logging.

`SETLOGLEVEL` - sets the system logging level.

`SETSHOWTIMES` - sets the system to display a full report of activity times on the console

`SETINTERNALDIAGNOSTICS` – a diagnostic function to print a diagnostic stack trace for every exception thrown (which optionally operates with `Log4J`)

### *Logging Functions*

TM provides a number of logging functions compatible with Java logging mechanisms such as `Log4J`, Java 1.4 logging, etc. By default, TM will use `Log4J` logging, but alternative logging mechanisms can be plugged in at run-time through the Java API or by specifying a system property, the latter being useful for setting the logger implementation used by the TM Test Tool or the TM Debugger.

Whichever implementation is used at run-time, the SML functions are used in the same way. There is no need for the transformation source code to know which logging mechanism will be used at runtime.

Note that if the SML explicitly requests logging to [StdOut](#) or [StdErr](#), then such log statements will always go to those streams. The pluggable log mechanism only relates to log statements that direct output to a named logger or the root logger.

The built-in logging functions are:

[LOGDEBUG](#) – Logs the supplied string at DEBUG level

[LOGERROR](#) – Logs the supplied string at ERROR level

[LOGEVENT](#) – Allows error information to be sent to a Log4J logger

[LOGFATAL](#) – Logs the supplied string at FATAL level

[LOGINFO](#) – Logs the supplied string at INFO level

[LOGWARN](#) – Logs the supplied string at WARN level

[SETLOGGINGCONFIG](#) – Allows you to configure the Log4J logging using a specified Log4J property configuration file

[SETLOGOUT](#) – Allows the user to re-direct existing messages generated by TM to a Log4J logger or a system output file

[STARTLOGCATEGORY](#) – Allows users to specify whether they want to log certain transformation categories, e.g. when source or target attributes are assigned, when entities are created or when errors are noted

[STOPLOGCATEGORY](#) – Stops the Log4J logging defined for a particular category using [STARTLOGCATEGORY](#)

## Error Handling

Errors in TM are of two main types:

- **System errors**
- **User errors**

### System Errors

A system error is generated whenever an error is detected accessing the source or target instances, such as if a database constraint is violated or if an XML node is not located. The source of a system error is therefore not evident by simply reviewing the transformation syntax.

### User Errors

A user error, on the other hand, is generated whenever any one of the SML transformation statements, [NOTEERROR](#), [THROWERROR](#) or [FATALERROR](#), is executed. Each type of SML error statement indicates the severity of the error and subsequent action to be taken.

All Errors in TM are written to the built-in model [TMErrors](#). [TMErrors](#) is included as the node [\\$TMErrors](#) in all repositories and is shown as part of both the source and target data model. Relationships can be set up from any element in the source or target model to [\\$TMErrors](#). Errors in TM are then processed during a number of stages, as shown in the figure below.

First, a [CATCH\\_ERROR](#) block, if present, may add further context to the error.

The next two stages are dependent upon the further processing that is required.

TM Errors are *unhandled* if no error handling projects exist and in this case the error may be written to either *StdOut*, *StdErr* or system file. Otherwise, the *handled* error will be processed by the Error Handling Projects (see further below) and may be output to a variety of targets, depending upon the user requirements.

The final stage, error strategy, is determined by the user error severity - whether to abort the transform (Fatal Error), to abort the current source instance (Throw Error), or to continue with the current source instance (Note Error). At this stage a system error is, by default, treated the same as a User Throw Error and processing continues with the next source instance.

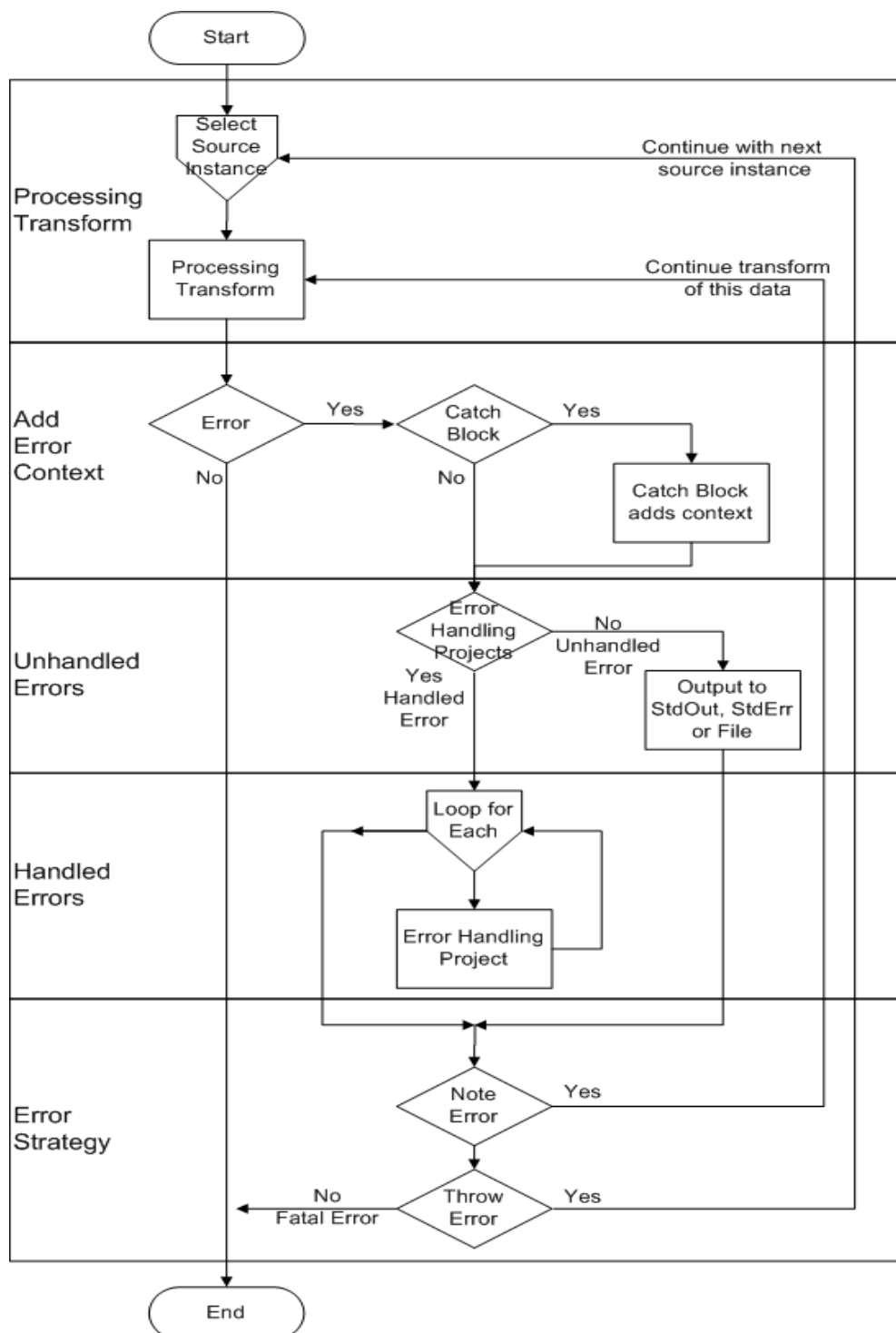


Figure 1 – the TM error processing stages



### ***Error Handling Projects***

A Transform Project can include any number of error handlers. The source of an error handling project is always the standard pre-loaded model called **TMEError**. The target can be any target supported by TM (e.g. XML, Java, RDBMS). All properties of the transformation system are available in an error handler (including the ability to catch and transform an error in the error handler itself).

An error handling project is expected to include one or more independent (and non-dependent) transforms defined from TMEError to items chosen in the target. All error handling projects included in the main project will be invoked in response to any error occurring in the main project. Errors may also occur inside the error handling projects. If this happens then the error is only handled by the other defined error handling projects. If no other error handling projects are available then the error will be regarded as unhandled and appear on a standard log output.

### **Further Information**

Website : [www.etlsolutions.com](http://www.etlsolutions.com)  
Email : [info@etlsolutions.com](mailto:info@etlsolutions.com)  
Telephone : +44 (0) 1912 894040