Log System

Administrator Manual

Software Version 1.2.0

EducateIT

EducateIT GmbH
Steinenbergstrasse 7, 4532 Feldbrunnen, Switzerland
Phone: +41 32 355 39 00
E-Mail: info@educateit.ch
Web: http://www.educateit.ch/

Log System – Administrator Manual
Copyright: ©2013 by EducateIT GmbH

EducateIT GmbH
Steinenbergstrasse 7
CH-4532 Feldbrunnen
Switzerland

Phone: +41 32 355 39 00
E-Mail: info@educateit.ch
Web: http://www.educateit.ch/

Illustrations and Text: TE
Layout and Design: TE
Last Modification: 2013-10-11 15:46

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by and means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the EducateIT GmbH. The EducateIT GmbH assumes no responsibility for errors or omissions, or for damages that may result from the use of information (especially including example configurations and listings) contained herein.

Content

Preface 5
Features 5
See Also 5

The Log System 7
Messages 7
Levels 7
Sources 8
Time Stamp 9
Message Text 9
Level Filter 10

Configuration **11**

Basic Structure 13

The `FileWriter` List 17
The `LogFilePath` Value 17
The `SourcePath` Value 17
The `LevelFilter` Value 18
The `MaximumSize` Value 18
The `Rotation` Value 19
The `LineFormat` Value 19
The `TruncateFile` Value 21

The `Filter` List 23
The `SourcePaths` Value 23
The `LevelFilter` Value 24

Additional Values 25
The `CreateConsoleWriter` Value 25

Preface

The "*EducateIT Log System*" is embedded in the software you got from our company. The system allows the software to write log messages in an efficient and fast way. For you as the user of the system, it provides a powerful configuration to specify log files and a format. It also provides options to create a new log file after a given amount of time to limit the size.

The first version of this system has a limited number of features. Be aware that later versions of the system may contain more features and functions you are probably interested in. Make sure you read the manual of a later version of the shipped software if the version of this system changed.

Features

- Fast log mechanism for the software.
- Stable and efficient.
- Simple configuration to write log files.
- Different log levels and logger hierarchy to filter messages of no interest.
- Trace sections to enable additional logged information for rare used areas or for generated information which could slow down the system.
- Log file rotation based on time.

See Also

The log system uses the *Configuration B* system for the configuration. To understand all aspects of the configuration, it is recommended to read the document "*Configuration B - Administrator Manual*". You should get this document along with this manual. If not, please request it from us.

The Log System

Writing log messages from system has many benefits:

- Events are recorded in chronological order.
- Network access for example is documented for later review.
- Problems and warnings are reported to the administrator of the system who can take action to solve the problem.
- Developers can trace problems to the source, checking the log entries of a problem case.

The *EducateIT Log System* was created with all of these uses in mind.

Messages

In case of an event a *log message* is sent to the log system. Each log message gets a *time stamp* with the current time and date. It contains the *message text* itself and a *log level*.

These messages are collected and can be sent to various *log writers* in the system. See Illustration 1 on this page for the different parts of a log message.

Levels

Each message is sent with a given *log level*. This level is an indicator of the importance of a message. Illustration 2 on the next page shows the log levels and their importance.

Usually messages with “information” character are the ones you always like to have in a log file. They contain for example the start and end times of important components or of the application itself, or they contain incoming connections.

Illustration 1 The different parts of a log message.

Messages with “errors” are very important messages as well. But probably you like to keep the main log file free from error messages and write them into a second log file.

“Warnings” are not as important as “errors”. You probably can even ignore those, they should not be a serious problem for a running system, but are strong indicators for problems with the configuration or the system.

And the least important log level is the “trace” level, which is practically only used by the developer to find problems in the software. You should filter them for productive systems.

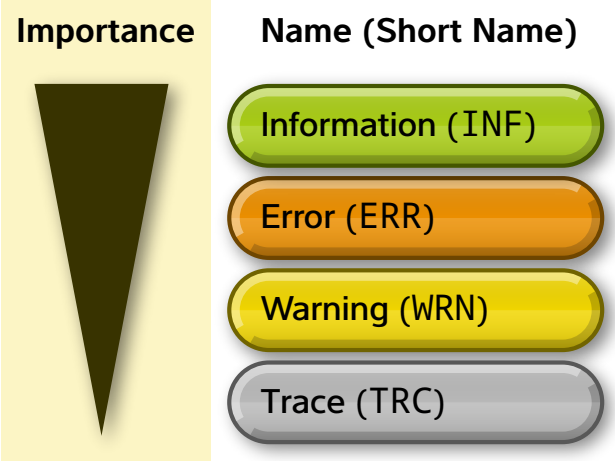

Illustration 2 The log levels and their importance.

The “importance” of these levels is used if you apply *filters* to *log sources* or *log writers*. They are explained in a later section in this manual.

Sources

An application usually consists of many components which can contain nested smaller components and so on. But log messages are generated at many different places. To organize the messages generated at all this places, each message has a *source path*. This source path is a logical grouping of all log messages done by the developer.

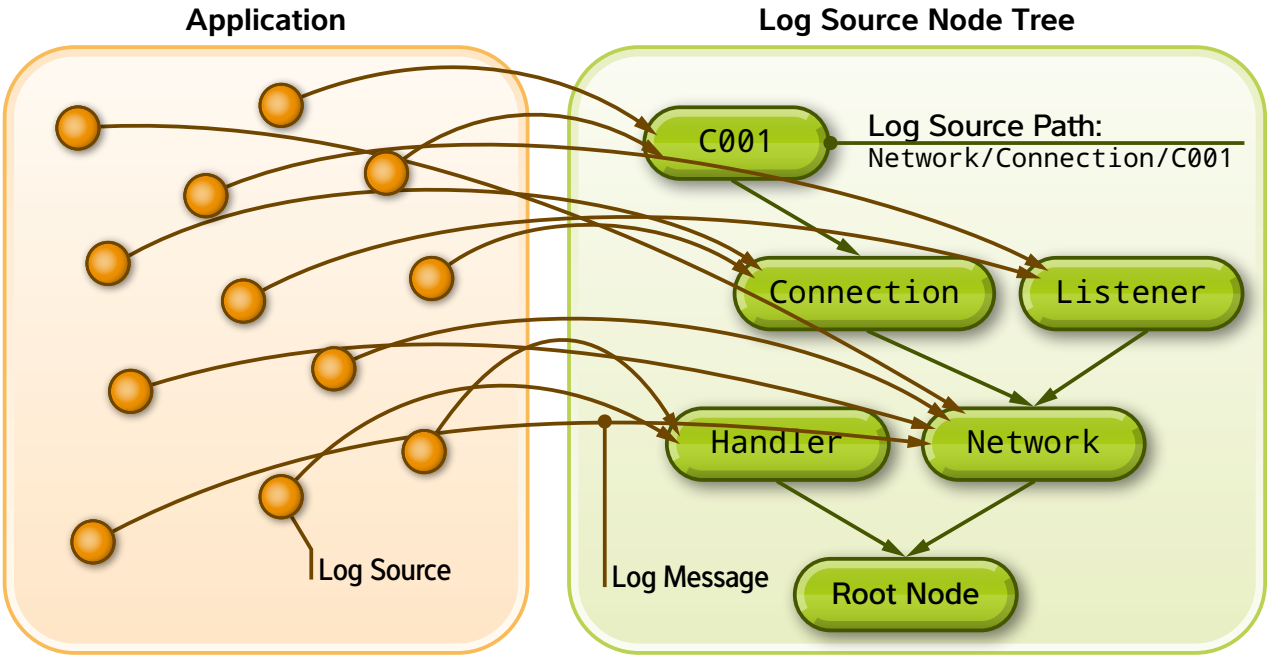

Illustration 3 Log message sources and the source node tree.

Each element of a such source path is building a node in the log system wide “source node tree”. This tree has a root node which is the base for each source path. Nodes at the same level with the same name are merged together. By processing all source paths in the application, a tree of nodes is created – all based on a single root node.

Have a look at Illustration 3 on the opposite page. It shows the application with all different log sources on the left side. Each of these sources is bound to a node in the log source node tree on the right side. There is one root node for the whole log system. The root node is always represented by an empty source path.

To build the source path, the names are read from the root node up to the node where the log source is attached to. A slash character separates the names. The source path of the top most node is shown in the illustration.

Time Stamp

The time stamp is a snapshot of the current date and time when the log message was created. While processing and writing of a log message can take some time, the time stamp always represents the time of creation.

This can lead to confusing situations on systems with a heavy load. Because this log system favours speed before accuracy while processing log messages, it is possible that the written log entries are not in the exact order of the time stamp. If this is a requirement for your project you have to post process the log files sorting the log messages by time.

Just to make things clear: I am referring here to disorder in the range of milliseconds, not seconds or even minutes. If you are only interested in the rough order of seconds or even minutes, this will never be a problem for you.

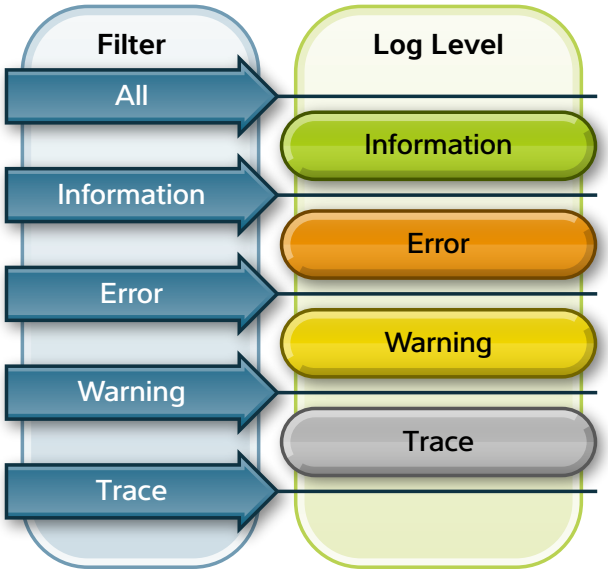

Illustration 4 The different filter level.

Message Text

The content of each message is just a line of text. This text has no fixed format, nor any restrictions.

Be aware that the whole log system is using Unicode text. The log message therefore can contain any Unicode character. This is the reason why log files of the system are encoded as UTF-8 by default.

Level Filter

To filter messages based on the log level you have to use a log level filer. There are five different possible values for a level filter which are shown in Illustration 4 on the previous page. To keep the naming as similar as possible with the level itself, they express the sentence “Filter everything below ...”. They only exception is the filter “All”, which filters all log messages.

See Illustration 5 on this page for four examples of how the filter value affects the different log level. The transparent log levels are filtered out.

The default filter in the configuration is always “Trace” which allows all log messages passing through.

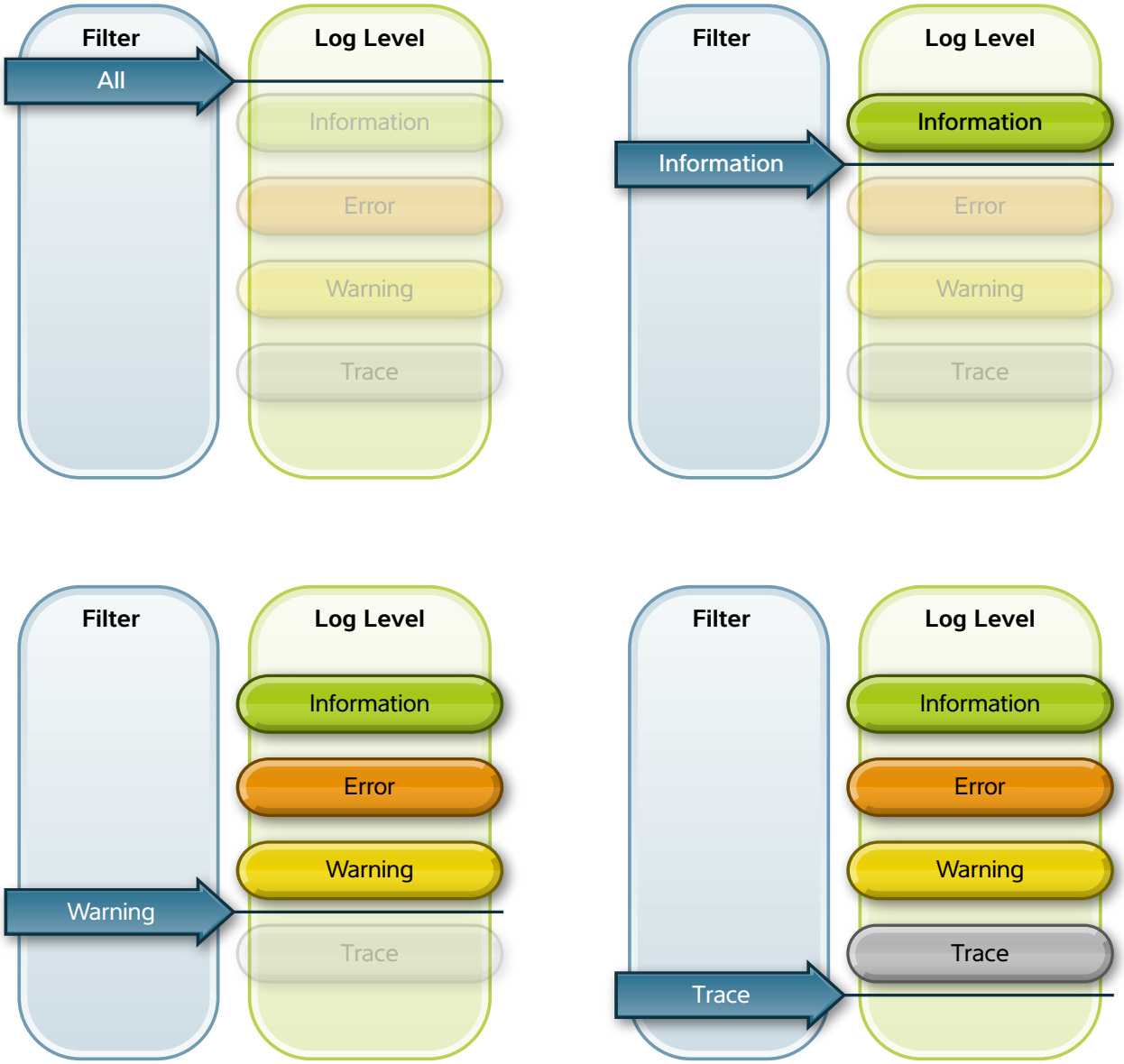

Illustration 5 Four examples how the filter value affects the log messages.

Chapter 1: Configuration

Basic Structure

Each software which uses the *Log System* along with the *Configuration B* system is supporting the "Log" module in the configuration file. This module is optional. If there is no such module defined in the configuration, logging is disabled for the given software.

Listing 2 on page 15 shows the structure of the log module in an application configuration and Illustration 6 on the next page shows the visualization of the configuration module. Most of the values are optional and have well defined defaults. For this reason, the log configuration is in the most cases very simple.

See Listing 1 on this page for a minimal example. This example writes all log messages into the specified file.

Listing 3 on page 15 shows a realistic configuration of the log system. It defines a log file which is rotated daily. It also limits the log up to warning messages.

The lists and values of the shown structure are explained in detail in the following sections.

```
<Module name="Log">
  <List name="FileWriter">
    <ListEntry>
      <Value name="LogFilePath">C:\LogFiles\project.log</Value>
    </ListEntry>
  </List>
</Module>
```

Listing 1 Simple configuration with one file writer.

Module	Log
Value	`CreateConsoleWriter` — Optional — Boolean
Default	`Yes`
List	`EnabledTraceSections` — Optional
Definition	Value List Definition
Constraints	▪ Has to match the following **case insensitive** regular expression: `\w+`
List	`FileWriter` — Optional
Definition	List Entry Definition
Value	`LevelFilter` — Optional — String
Constraints	▪ Has to be one of the following values: `All`, `Information`, `Error`, `Warning`, `Trace`
Default	`Trace`
Value	`LineFormat` — Optional — String
Value	`LogFilePath` — String
Value	`MaximumSize` — Optional — Integer
Default	`-1`
Value	`Rotation` — Optional — String
Constraints	▪ Has to be one of the following values: `NoRotation`, `Hourly`, `Daily`, `Weekly`, `Monthly`
Default	`NoRotation`
Value	`SourcePath` — Optional — String
Value	`TruncateFile` — Optional — Boolean
Default	No
List	`Filter` — Optional
Definition	List Entry Definition
Value	`LevelFilter` — String
Constraints	▪ Has to be one of the following values: `All`, `Information`, `Error`, `Warning`, `Trace`

Illustration 6 The Log module configuration.

```
<Module name="Log">
  <List name="FileWriter">
    <ListEntry>
      <Value name="LogFilePath">C:\LogFiles\project.log</Value>
      <Value name="LevelFilter">Warning</Value>
      <Value name="Rotation">Daily</Value>
    </ListEntry>
  </List>
</Module>
```

Listing 3 Realistic configuration of the log module.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Configuration
  version="1"
  xmlns="http://educateit.ch/software/BlueStone/Configuration/1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Module name="Application">
    ... application configuration ...
  </Module>

  ... additional modules ...

  <Module name="Log">

    <List name="FileWriter">
      <ListEntry>
        <Value name="LogFilePath">...</Value>
        <Value name="SourcePath">...</Value>
        <Value name="LevelFilter">...</Value>
        <Value name="MaximumSize">...</Value>
        <Value name="Rotation">...</Value>
        <Value name="LineFormat">...</Value>
        <Value name="TruncateFile">...</Value>
      </ListEntry>
      ...
    </List>

    <List name="Filter">
      <ListEntry>
        <Value name="SourcePaths">...</Value>
        <Value name="LevelFilter">...</Value>
      </ListEntry>
      ...
    </List>

    <Value name="CreateConsoleWriter">...</Value>

  </Module>

</Configuration>
```

Listing 2 Structure of the log module in a configuration file.

The FileWriter List

This is an optional list which defines one or more log files to write. Each entry requires an absolute path to the log file which is specified in the `LogFilePath` value. All other values are optional and have well defined defaults.

The log file is always encoded as UTF-8 using

The LogFilePath Value

This is the only required value for a log file writer entry. It has to be an absolute path to the log file. All directories to the file have to exists. You should also specify a suffix for the file, like "`.log`" or "`.txt`".

For Windows, it is not necessary to use the backslash as path separator. You can use the slash like on all other operating systems.

The following listing shows an example how the log file path can be defined:

```
<ListEntry>
  <Value name="LogFilePath">e:/Test/application.log</Value>
</ListEntry>
```

This example is also a valid entry to write a log file. All other values are optional and can be omitted.

The SourcePath Value

Inside of each software are a number of log sources. If you specify a source path to a source node with this value, this log file writer is limited to log output from the given source node and sub nodes.

The default is an empty value, which addresses the root source node and therefore logs the messages from the whole system.

In the example below, only messages from the component manager are logged:

```
<ListEntry>
  <Value name="LogFilePath">e:/Test/application.log</Value>
  <Value name="SourcePath">dzcore/ComponentManager</Value>
</ListEntry>
```

If there were source nodes below the component manager, for example in the path "`dzcore/ComponentManager/example`" these message were written as well.

The source path must not start or end in a slash character, and must not contain whitespace.

The `LevelFilter` Value

With this value you can set a filter for the log writer. The different filter values are explained in "Level Filter" on page 10. This filter affects only the messages written to the file defined in this entry. You also can not undo the effect of a filter you already have put on a log source node.

The following example writes only "Information" and "Error" messages into the log file:

```
<ListEntry>
  <Value name="LogFilePath">e:/Test/application.log</Value>
  <Value name="LevelFilter">Error</Value>
</ListEntry>
```

The `MaximumSize` Value

This optional value sets an maximum size of the log file. If you set a maximum size, the log file is rotated when it exceeds the given size. The given size is in bytes, but can add the suffix "`kb`", "`mb`" or "`gb`" to specify kilobytes, megabytes or gigabytes.

The special value `-1` disables the rotation based on size. This is the default if you omit this value.

In the following example, the log file is rotated when it exceeds 120 megabytes:

```
<ListEntry>
  <Value name="LogFilePath">e:/Test/application.log</Value>
  <Value name="MaximumSize">120mb</Value>
</ListEntry>
```

If this option is enabled by setting a limit, the log system adds a suffix to the file name. The example above would generate names as shown in the next block:

```
application-00001.log
application-00002.log
application-00003.log
...
```

Rotation	Pattern	Example for file "`application.log`"
`Hourly`	`_yyyymmdd_hh`	`application_20120501_10.log`
`Daily`	`_yyyymmdd`	`application_20120501.log`
`Weekly`	`_yyyy_ww`	`application_2012_17.log`
`Monthly`	`_yyyymm`	`application_201205.log`

Table 1 The suffixes added to the log file name, depending on the "`Rotation`" value.

The `Rotation` Value

This optional value enables time based rotation of the log file. You can choose "`Hourly`", "`Daily`", "`Weekly`" and "`Monthly`". The special value "`NoRotation`" disables the time based rotation. This is the default if you omit this value.

If rotation is enabled for a log file, the system adds a suffix to the end of the file name. Table 1 on the opposite page is showing the pattern of the suffix and an example how the name of a log file could look like using this pattern.

Be aware that rotating a log file is not done at the exact millisecond when the time or date changes. It is always possible that there is one or more message left in the old log file which would belong to the rotated file. This behaviour exists, because the log system favours speed before accuracy.

The `LineFormat` Value

This optional value defines the format of the line which is written to a log file. The format is a string which can contain any characters and special placeholders which are replaced by the different attributes of a log message.

The following example shows the default format for a log line, expressed as a line format text:

```
{level} {datetime} {source} {message}
```

Using this format, a written log line can look like this:

```
INF 2012-07-06 14:57:49 dzsvc/Application Starting App...
```

Everything outside the placeholder is copied without modification into the resulting line. In our example above, these are just the spaces between the placeholder.

The Format of a Placeholder

Each placeholder is enclosed into curly brackets. It starts with the placeholder identifier. For example "`level`" or "`datetime`". Some placeholder are supporting additional parameter, they are separated with the pipe character ("`|`"). Each parameter starts with the parameter name, followed by an equal sign, which is followed by the parameter value.

The next example shows the basic format of a parameter:

```
{name}                                  ← Simple
{name|paramName=valie}                  ← One Parameter
{name|p1=value|p2=value|p3=value}       ← Multiple Parameter
```

The `level` Placeholder

This placeholder is replaced by the level of the log message. It is supporting two parameter to control the output format for the level. The parameter "`case`" changes the use of upper and lower case letters and the parameter "`format`", which changes the way that the level is written to the log.

Table 2 on this page shows combinations of the two parameters, and how the "Information" and "Warning" level is rendered. The line in orange with the bold text is the default for these parameters if you omit them.

```
{level}
{level|case=Upper|format=Short1}
```

The `datetime` Placeholder

Use this placeholder to add the time and date to the log file. It has one parameter "`format`" where you can define the format of the date and time.

The date and time format is defined by using one or more letter combinations explained in Table 3 on the opposite page. All other characters are just treated as text. Any sequence of characters enclosed in single quotes are treated as text.

```
{datetime}
{datetime|format=yyyyMMdd hhmmss}
```

The `source` Placeholder

This placeholder inserts the source path of the message. There are no parameters for this placeholder.

case	format	Rendering of "Information"	Rendering of "Warning"
Mixed	Full	Information	Warning
Upper		INFORMATION	WARNING
Lower		information	warning
Mixed	Short1	I	W
Upper		I	W
Lower		i	w
Mixed	**Short3**	**INF**	**WRN**
Upper		INF	WRN
Lower		inf	wrn

Table 2 Combinations of the two parameter of the `level` placeholder and the result.

The `message` Placeholder

Use the message placeholder to insert the log message itself into the log line. This placeholder has no parameter.

The `TruncateFile` Value

This value is usually just used for testing. If the value is set to `Yes`, each file is “truncated” (reset to zero length) before any writing begins. The default value is `No` and in a productive environment you should omit this value or set it to `No`.

Expression	Description
`d`	The day as number without a leading zero (1 to 31).
`dd`	The day as number with a leading zero (01 to 31).
`ddd`	The abbreviated localized day name (for example ‘Mon’ to ‘Sun’).
`dddd`	The long localized day name (for example ‘Monday’ to ‘Sunday’).
`M`	The month as number without a leading zero (1-12).
`MM`	The month as number with a leading zero (01-12).
`MMM`	The abbreviated localized month name (for example ‘Jan’ to ‘Dec’).
`MMMM`	The long localized month name (for example ‘January’ to ‘December’).
`yy`	The year as two digit number (00-99).
`yyyy`	The year as four digit number.
`h`	The hour without a leading zero (0 to 23 or 1 to 12 if AM/PM display).
`hh`	The hour with a leading zero (00 to 23 or 01 to 12 if AM/PM display).
`H`	The hour without a leading zero (0 to 23, even with AM/PM display).
`HH`	The hour with a leading zero (00 to 23, even with AM/PM display).
`m`	The minute without a leading zero (0 to 59).
`mm`	The minute with a leading zero (00 to 59).
`s`	The second without a leading zero (0 to 59).
`ss`	The second with a leading zero (00 to 59).
`z`	The milliseconds without leading zeroes (0 to 999).
`zzz`	The milliseconds with leading zeroes (000 to 999).
`AP or A`	Interpret as an AM/PM time. AP must be either “AM” or “PM”.
`ap or a`	Interpret as an AM/PM time. ap must be either “am” or “pm”.

Table 3 A table with all valid date and time format expressions.

The Filter List

This optional list defines one or more filters on selected source paths. Each entry on the list has two required values. The "SourcePaths" value which defines a list of source paths to apply the filter to, and the "LevelFilter" value which filters messages based on the message level.

Illustration 7 on this page shows the log system with five different messages on the left, sent to different nodes in the system in the middle. All messages which reach the log writer end up in the log file.

In Illustration 8 on the next page an "Error" level filter is applied on the "Network/Connection" source path. And in Illustration 9 on the next page the level filter is set to "All".

The SourcePaths Value

This value needs one or more source paths, separated with whitespace. The filter is applied to all this source paths.

```
<ListEntry>
  <Value name="SourcePaths">Network/Connection</Value>
  ...
</ListEntry>
```

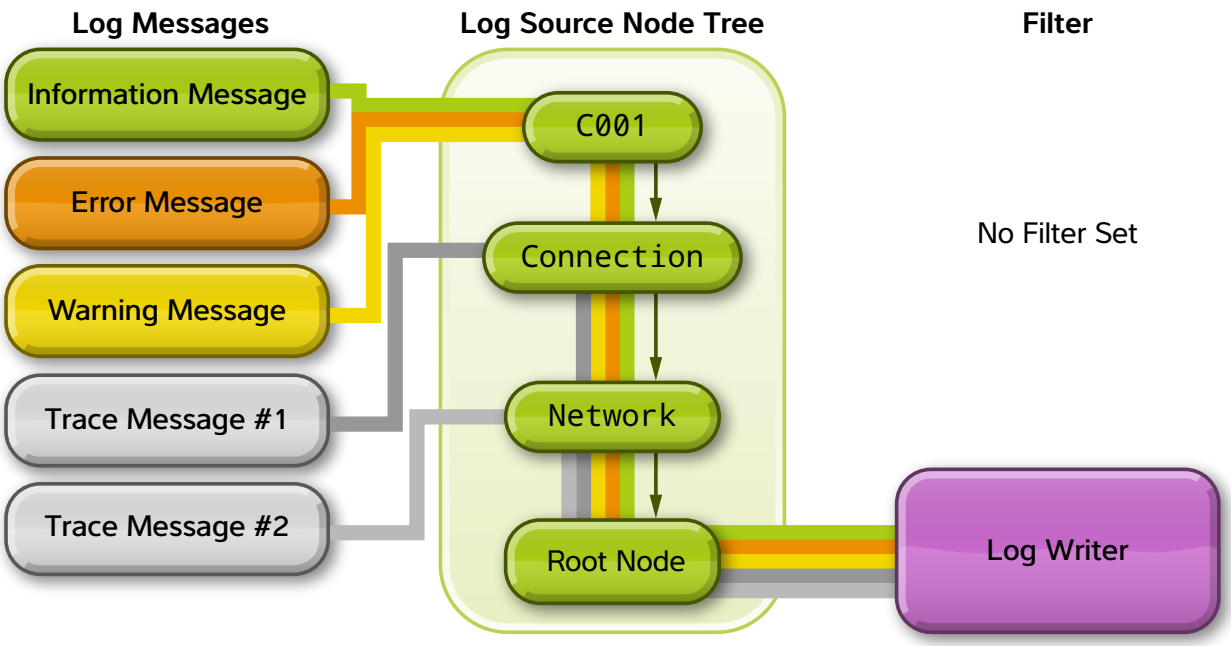

Illustration 7 Log system with five messages and without a filter.

The LevelFilter Value

This value sets which levels shall be filtered at the selected source nodes. The meaning of the filter value is explained in section "Level Filter" on page 10 in detail.

```
<ListEntry>
  ...
  <Value name="LevelFilter">Error</Value>
</ListEntry>
```

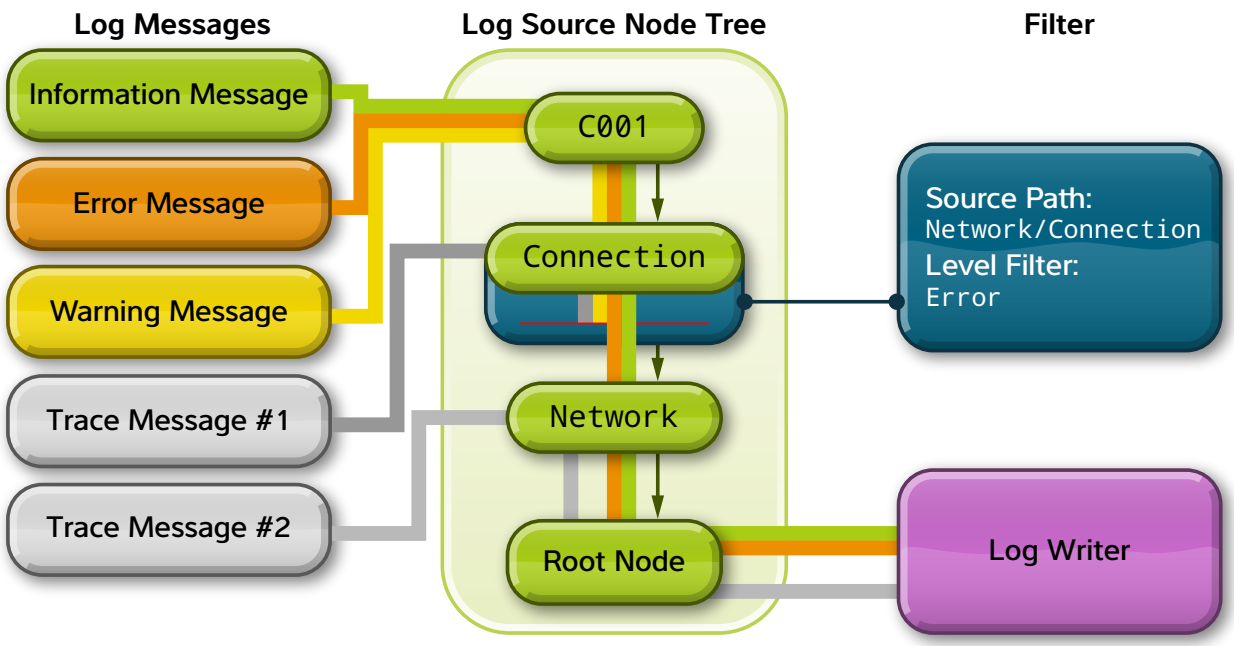

Illustration 9 Log system with five messages and with an "Error" filter on one node.

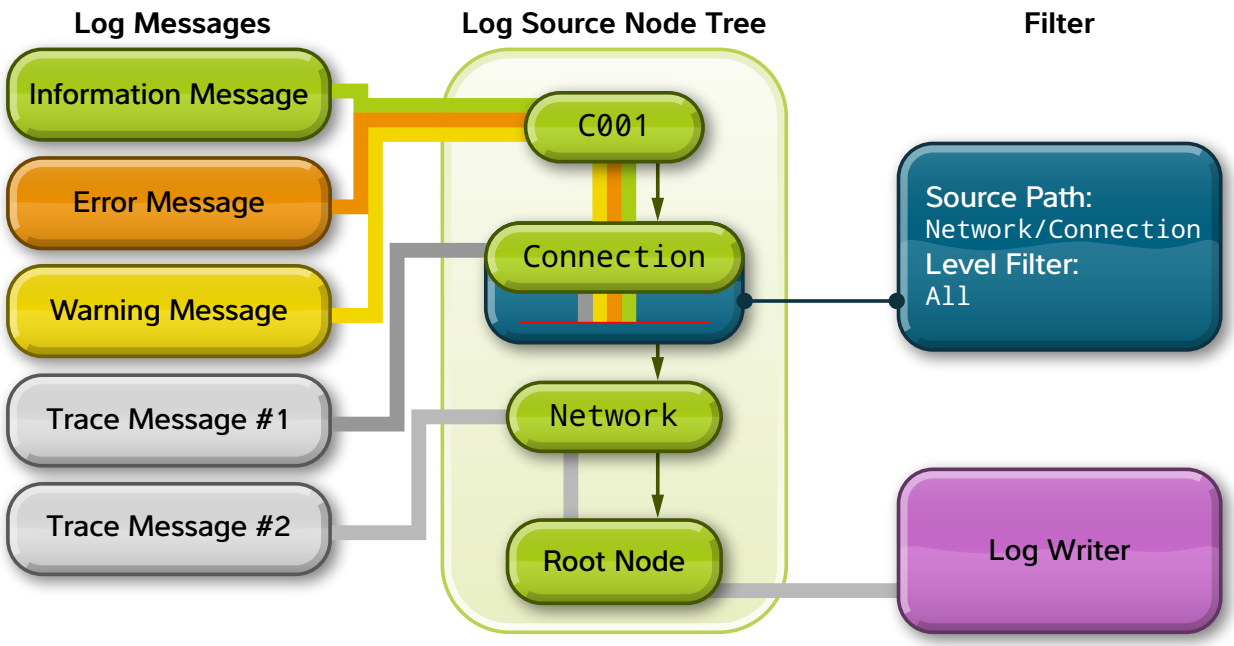

Illustration 8 Log system with five messages and with the level filter "All" on one node.

Additional Values

This section describes additional values of the configuration system. Currently there is only one values.

The `CreateConsoleWriter` Value

This value is only used for testing. It controls if a console writer should be created at the application start. You are advised to omit this value for productive environments.

The `EnabledTraceSections` List

You can enable trace sections with this optional list. Each entry in this list enables a given trace section. This feature is only used for testing. In a productive environment no trace sections shall be enabled.

```
<Module name="Log">
    <List name="FileWriter">
      <ListEntry>
        <Value name="LogFilePath">C:/application.log</Value>
      </ListEntry>
    </List>
    <List name="EnabledTraceSections">
      <Value>msgKeepAlive</Value>
    </List>
  </Module>
```

EducateIT GmbH
Steinenbergstrasse 7, 4532 Feldbrunnen, Switzerland
Phone: +41 32 355 39 00
E-Mail: info@educateit.ch
Web: http://www.educateit.ch/