Log4Cplus Usage

Introduction

Let's consider configuring FIXAntenna since version 2.26 or FIXEdge since version 6.7 to send logs to the Splunk or Splunk Agent running on the localhost and listening on port 1514. The example configuration is as follows:

Example : Forwarding logs to the Splunk

```
# add Log4Cplus device for forwarding logs to the log4cplus
Log.Device = File Log4Cplus

#------------- configure log4plus -------------------------
log4cplus.rootLogger = TRACE,Splunk
log4cplus.appender.Splunk=log4cplus::SocketTCPAppender
#set host/port Splunk
log4cplus.appender.Splunk.port=1514
log4cplus.appender.Splunk.host=localhost
# using pattern for add information in log messages about
log4cplus.appender.Splunk.layout=log4cplus::PatternLayout
log4cplus.appender.Splunk.layout.ConversionPattern=%d{%FT%T.%q}Z Severity=%-5p ThreadID=%t Category=%c %m%n
```

This configuration should be added to `engine.properties` (for FIXAntenna) or `FIXEdge.properties` (for FIXEdge). Here logs are sent to the two devices: `File` to write to a file on the local filesystem and `Log4Cplus` to send logs to Splunk.

Loggers

Logger name are identical to the Log Category. By default messages are processed to the logger with the Log Category and to the root logger. To configure the logging subsystem:

1. ***Set the root logger. The root logger can be assigned to a logging level and one or more formatting handlers.***

   The syntax for configuring the root logger is the following:

engine.properties (for FIXAntenna) or FIXEdge.properties (for FIXEdge)

```
log4cplus.rootLogger=[LogLevel], appenderName, appenderName, ...
```

Where

- **[LogLevel]** is an optional parameter and can consist of the string values (**Logging levels**):

  FATAL: Logs very severe error events that may lead the application to abort.

  ERROR: Logs only error conditions. The ERROR level provides the smallest amount of logging information.

  WARN: Logs information when an operation completes successfully but there are issues with the operation.

  INFO: Logs information about workflow. It generally explains how an operation occurs.

  DEBUG: Logs all of the details related to a specific operation. This is the highest level of logging.

  TRACE: Logs designated finer-grained informational events than DEBUG.

- If a **LogLevel** value is specified, then the root **LogLevel** is set to the corresponding **LogLevel**. If no **LogLevel** value is specified, then the root **LogLevel** remains untouched.
- **appenderName** contains the information on where to redirect the logging output (for example, console, file, syslog, etc.). The root logger can be assigned to the multiple appenders. Each **appenderName** (separated by commas) will be added to the root logger.

Example:

engine.properties (for FIXAntenna) or FIXEdge.properties (for FIXEdge)

```
#configure root logger for redirected to TCP socket
log4cplus.rootLogger = TRACE,ServerTCP
#configure appender  redirected to socketTCP
log4cplus.appender.ServerTCP=log4cplus::SocketTCPAppender
log4cplus.appender.ServerTCP.port=1514
log4cplus.appender.ServerTCP.host=localhost
```

2. ***Set non-root loggers (optional step).***

engine.properties (for FIXAntenna) or FIXEdge.properties (for FIXEdge)

```
log4cplus.logger.logger_name=[LogLevel], appenderName, appenderName, ...
```

Example:

FIXEdge.properties

```
#configure Engine logger for redirected to console
log4cplus.logger.Engine = ERROR,ErrorsEngine
#configure appender
log4cplus.appender.ErrorsEngine=log4cplus::ConsoleAppender
```

3. ***Disable the duplication of the messages to the root logger (optional step).***

This option allows to avoid duplication to the root logger.

```
logger.additivity.<nameOfLogger> = false
```

Default value: logger.additivity.<nameOfLogger> = true

Appenders

log4cplus::NullAppender

Provides an option to discard an output.

Example:

```
#configure root logger
log4cplus.rootLogger = TRACE,OutputToFile
#configure appender for output errors
log4cplus.appender.OutputToFile = log4cplus::NullAppender
```

log4cplus::SocketTCPAppender

Provides an option to redirect an output to the TCP socket.

Property	Description	Default	Example
host (required)	Server host		127.0.0.1

port (required)	Server port		1514
sendTimeout	Timeout on send data in socket (SO_SNDTIMEO) in microsecond. After this connection will be considered broken and begin reconnection Valid values : 1. 0 - will not use timeout . 2. Min - 1024, Max - 2147483647 (On linux maximum is limited of linux /proc/sys/net/core/wmem_max), Recommended - default value	100000 (100 millsecond)	
label	In error will add label : log4cplus::SocketTCPAppender::<label>(localhost:1514) : Connection to server established. Valid values: 1. empty 2. any text	empty	

Layouts

Provides an option to change messages in appender. To configure usage, add property "*layout*" for appender.
Example :

Example:

```
log4cplus.appender.OutputToFile.layout=log4cplus::PatternLayout
```

log4cplus::PatternLayout

A flexible layout configurable with pattern string. For more information please refer to http://log4cplus.sourceforge.net/docs/html/classlog4cplus_1_1PatternLayout.html.

The goal of this class is to format a InternalLoggingEvent and return the results as a string. The results depend on the *conversion pattern*.

The conversion pattern is closely related to the conversion pattern of the printf function in C. A conversion pattern is composed of literal text and format control expressions called *conversion specifiers*.

Please note that any literal text can be used within the conversion pattern.

ConversionPattern

Each conversion specifier starts with a percent sign (%%) and is followed by optional *format modifiers* and a *conversion character*. The conversion character specifies the type of data, e.g. Logger, LogLevel, date, thread name. The format modifiers control such things as field width, padding, left and right justification.

Example:

```
log4cplus.appender.OutputToFile.layout=log4cplus::PatternLayout
log4cplus.appender.OutputToFile.layout.ConversionPattern=%d{%FT%T.%q}Z Severity=%-5p ThreadID=%t Category=%c %m%n
```

Please find the recognized conversion characters in the table below:

Conversion Character	Effect
c	Category LogSystem Used to output the logger of the logging event. The logger conversion specifier can be optionally followed by *precision specifier*, that is a decimal constant in brackets. If a precision specifier is given, then only the corresponding number of right most components of the logger name will be printed. By default the logger name is printed in full. For example, for the logger name "a.b.c" the pattern c{2} will output "b.c".

d	Used to output the date of the logging event in UTC. The date conversion specifier may be followed by a *date format specifier* enclosed between braces. For example, %d{%H:%M:%s} or %d{%d %b %Y %H:%M:%s}. If no date format specifier is given then %d{%d %m %Y %H:%M:%s} is assumed. The Following format options are possible: %a – Abbreviated weekday name %A – Full weekday name %b – Abbreviated month name %B – Full month name %c – Standard date and time string %d – Day of month as a decimal(1-31) %H – Hour(0-23) %I – Hour(1-12) %j – Day of year as a decimal(1-366) %m – Month as decimal(1-12) %M – Minute as decimal(0-59) %p – Locale's equivalent of AM or PM %q – milliseconds as decimal(0-999) – *Log4CPLUS specific* %Q – fractional milliseconds as decimal(0-999.999) – *Log4CPLUS specific* %S – Second as decimal(0-59) %U – Week of year, Sunday being first day(0-53) %w – Weekday as a decimal(0-6, Sunday being 0) %W – Week of year, Monday being first day(0-53) %x – Standard date string %X – Standard time string %y – Year in decimal without century(0-99) %Y – Year including century as decimal %Z – Time zone name %% – The percent sign Please refer to the documentation for the `strftime()` function in the <ctime> header for more information - http://man7.org/linux/man-pages/man3/strftime.3.html. Example for ISO 8601 : %d{%FT%T.%q}Z - 2018-04-03T09:03:44.806Z %d{%FT%T.%q}+00:00 - 2018-04-03T09:03:44.806+00:00
D	Used to output the date of the logging event in local time. All of the above information applies. Example for ISO 8601 : %D{%FT%T.%q%z} - 2018-04-03T09:03:44.806-0400 %D{%FT%T.%q}%.-3D{%z}:%.2D{%z} - 2018-04-03T09:03:44.806-04:00
E	Used to output the value of a given environment variable. The name of is supplied as an argument in brackets. If the variable does exist then empty string will be used. For example, the pattern E{HOME} will output the contents of the HOME environment variable.
F	Used to output the file name where the logging request was issued. *NOTE:* Unlike log4j, there is no performance penalty for calling this method.

h	Used to output the hostname of this system (as returned by gethostname(2)). *NOTE:* The hostname is only retrieved once at initialization.
H	Used to output the fully-qualified domain name of this system (as returned by gethostbyname(2) for the hostname returned by gethostname(2)). *NOTE:* The hostname is only retrieved once at initialization.
l	Equivalent to using "%F:%L" *NOTE:* Unlike log4j, there is no performance penalty for calling this method.
m	Used to output the application supplied message associated with the logging event.
n	Outputs the platform dependent line separator character or characters.
p	Used to output the LogLevel of the logging event.
r	Used to output miliseconds since program start of the logging event.
t	Used to output the thread ID of the thread that generated the logging event. (This is either `pthread_t` value returned by `pthread_self()` on POSIX platforms or thread ID returned by `GetCurrentThreadId()` on Windows.)
T	Used to output alternative name of the thread that generated the logging event.
i	Used to output the process ID of the process that generated the logging event.
"%%"	The sequence "%%" outputs a single percent sign.

Troubleshooting

Using SocketTCPAppender. Standard error output has errors on timeout sending

Error log:

```
log4cplus:ERROR log4cplus::SocketTCPAppender127.0.0.1:1514) : Timeout sending ( 100000 us ). Starting
reconnection to server. Next events will be skipped.
```

Description: Splunk server is a slow consumer. Appender disconnects connection after a timeout to avoid blocking threads of FIXEdge.

Solution: Set a larger value for property sendTimeout or use value '0' to switch off sending timeout.

Disabling timeout or large values for it means that logging has more priority than handling FIX traffic.