--------------------------------------------------------------------------

Syslog4j FAQ

Version: $Id: faq.html,v 1.17 2010/09/11 16:49:23 cvs Exp $

URL: http://www.syslog4j.org/

--------------------------------------------------------------------------

**1. General**

**1.1. What is Syslog4j?**

Syslog4j is a pure-Java implementation of the syslog client and server

specification as defined by "The BSD syslog Protocol" (RFC3164) with

additional support for TCP/IP-based syslog, native Unix syslog, and

Unix sockets.

Java applications (web or otherwise) needing to log messages to a syslog

server or Java applications needing to accept syslog messages can use

Syslog4j.

**1.2. Doesn't a Java syslog client already exist?**

Not really. Various Internet searches present a few options, but no

officially-supported solutions seem to exist (as of October 2008).

One solution is found at http://www.trustice.com/java/syslog/ but it

appears (as of this writing) that the code has not been updated since

2001, and attempting to download the distributions sometimes

results in errors.

JavaWorld has a posting about syslog from 2001 here:

http://www.javaworld.com/jw-04-2001/jw-0406-syslog.html

Several quick-and-dirty UDP-based code examples are also available via

Internet searches.

**1.3. Why use Syslog4j?**

It's easy and extensible!

Syslog4j provides UDP, TCP, Native Unix syslog, and Unix socket logging

with a simplified interface and is designed using an extensible coding

model that allows developers to improve upon or add new capabilities.

Of course, it's recommended any improvements are submitted to the Syslog4j

developers so Syslog4j can continually evolve.

**1.4. What dependencies are required for Syslog4j?**

Syslog4j requires JDK 1.4 or higher and requires NO EXTERNAL

LIBRARIES (JARs) for basic UDP/IP and TCP/IP clients and servers.

Unix syslog and/or Unix sockets require the JNA library,

available at: https://jna.dev.java.net/

Pooled connections for TCP/IP and TCP/IP over SSL/TLS

require the Apache Commons Pool library available

at: http://commons.apache.org/pool/

Log4j support requires, well, Apache Log4j, available at:

http://logging.apache.org/log4j/

**1.5. Is Syslog4j cross-platform?**

Yes! Syslog4j UDP/IP and TCP/IP clients should work in any

typical Java JRE environment.

**1.6. How do I use Native Unix syslog or Unix sockets?**

Because the Java JRE does not contain support for Unix syslog and

Unix socket logging, Syslog4j requires "hooks" into operating

system-specific libraries, and thus the JNA library is required.

The JNA library is available here:

https://jna.dev.java.net/

This was chosen in lieu of JNI, which requires operating system-

specific code to support the native library calls. JNA provides

a stand-alone JAR without any need for any additional libraries.

**1.7. Which license does Syslog4j use?**

Syslog4j is released as free software under the Lesser GNU Public

License v2.1, defined at the following URL:

http://www.gnu.org/licenses/lgpl-2.1.html

Additionally, the Open Source Initiative (OSI) lists the LGPL license in

its directory of accepted open source licenses, here:

http://www.opensource.org/licenses/lgpl-2.1.php

A copy of the LGPLv2 is made available in the META-INF directory of the

Syslog4j JAR distributions in a file called LICENSE.txt.

This license is used by a large number of library-oriented projects on

the Internet, including the largely popular "Hibernate" persistence

engine.

--------------------------------------------------------------------------

**2. Syslog4j Use & Examples**

**2.1. Where do I obtain Syslog4j?**

Syslog4j is available here:

http://www.syslog4j.org/

**2.2. There seem to be several choices for download. Which one do I use?**

Every release, there are four different JAR distributions and one

ZIP document distribution for Syslog4j:

Binary + Source Distribution: syslog4j-x.y.z.jar

Binary Distribution: syslog4j-x.y.z-bin.jar

Source Distribution: syslog4j-x.y.z-src.jar

JUnit (Test) Distribution: syslog4j-x.y.z-junit.jar

Documentation Distribution: syslog4j-x.y.z-doc.zip

To get started, it is recommended that you use the "Binary + Source"

distribution, as it (a) is compiled with the javac "debug" flag and

(b) allow you access to the Syslog4j source code for ease in debugging.

If you have concerns about performance or including source code in your

application, you're welcome to use the "Binary" Distribution. It is

compiled without the javac "debug" flag and the source code is not

included within the JAR file.

If you wish to compile your own Syslog4j from the source code, the

"Source" distribution is your best choice.

If you'd like to JUnit test Syslog4j or see working code examples,

the "JUnit (Test)" distribution will be of interest to you. Source

code for the JUnit tests are included in this distribution.

The "Documentation" distribution contains the API (JavaDoc), this

document (the FAQ), and other potentially useful information.

**2.3. How do I start using Syslog4j?**

For syslog clients, start with the singleton class called "Syslog"

(org.productivity.java.syslog4j.Syslog). This class provides access to

get existing instances and create new instances of Syslog4j "protocols."

For syslog servers, start with the singleton class called "SyslogServer"

(org.productivity.java.syslog4j.server.SyslogServer).

**2.4. What is a Syslog4j "protocol"?**

This is a String that defines the actual underlying Syslog4j

implementation you wish to use. Syslog4j comes "ready to run" with

the following stock protocols:

"udp" -- The original syslog "UDP" protocol, pointed to 127.0.0.1

(localhost), configured to use the "USER" syslog facility.

"tcp" -- An implementation of the de-facto TCP/IP syslog protocol,

which points to a TCP-capable syslog server, defaulted to

127.0.0.1 (localhost), configured to use the "USER" syslog

facility.

"unix\_syslog" -- Native syslog support for Unix platforms, configured

to use the "USER" syslog facility.

"unix\_socket" -- Native socket support for Unix platforms, pointed to

the "/var/log" socket, configured to use the "USER" syslog

facility.

**2.5. How do I use one of the stock protocols?**

There are two ways -- the "direct" way and the "via an interface" way, shown

below:

// Direct UDP Use Example

Syslog.getInstance("udp").info("This is an INFO level log entry.");

// Via an Interface UDP Use Example

SyslogIF syslog = Syslog.getInstance("udp");

syslog.info("This is another INFO level log entry.");

To use "tcp" protocol, simply switch the "udp" above to: "tcp"

**2.6. How do I point one of the stock protocols to go to a particular**

**host or port?**

There are two ways -- the "direct" way and the "via an interface" way, shown

below:

// Direct UDP Config Example

Syslog.getInstance("udp").getConfig().setHost("syslogserver.example.com");

Syslog.getInstance("udp").getConfig().setPort(9999);

// Via an Interface UDP Config Example

SyslogConfigIF config = Syslog.getInstance("udp").getConfig();

config.setHost("syslogserver.example.com");

config.setPort(9999);

**2.7. How do I send messages to more than one syslog server?**

The stock protocols in Syslog4j are designed as a convenience for those

that need to send to one server only. To point to more than one syslog

server, it's recommended that you create your own custom Syslog4j

protocol(s).

MultipleSyslog is an aggregate protocol that can be used to send

the same log entries to multiple stock or custom protocols.

**2.8. How do I create my own custom Syslog4j protocol?**

Syslog4j uses the coding paradigm of "dependency injection." In this

model, you create a Syslog4j configuration object and feed this object

to the Syslog.createInstance(protocol,config) method.

Here's an example, creating a new "audit" protocol which uses a UDP

implementation under the scenes:

// Syslog4j Protocol Configuration Creation Example

SyslogConfigIF config = new UDPNetSyslogConfig();

config.setHost("auditsyslog.example.com");

config.setFacility("LOCAL7");

// Syslog4j Instance Creation Example

Syslog.createInstance("audit",config);

This new "audit" instance is pointed to a specific server -- in

this case "auditsyslog.example.com" -- and is configured to use

the "LOCAL7" syslog facility.

Since you've created a new instance, the original stock "udp"

instance is not affected and can still be used.

**2.9. How do I use my own custom Syslog4j protocol?**

The Syslog.createInstance(protocol,config) method returns the

SyslogIF implementation if you'd like to use the Syslog4j right away:

// Syslog4j Instance Creation Example

SyslogIF syslog = Syslog.createInstance("audit",config);

syslog.info("This is the first INFO entry to this Syslog4j protocol.");

But normally, you'll be calling Syslog4j from other areas of your

code, so use the "direct" or "instances" ways as previously described:

// Custom UDP Syslog4j Protocol Direct Example

Syslog.getInstance("audit").info("This is an INFO entry");

// Custom UDP Syslog4j Protocol Instance Example

SyslogIF syslog = Syslog.getInstance("audit");

syslog.info("This is an INFO entry to the auditsyslog server");

**2.10. How do I close a client protocol connection?**

UDP/IP-based, Unix socket, and Unix syslog client protocols do not need

to be explicitly closed.

TCP/IP-based (including SSL/TLS) can be explicitly closed by calling

the flush() method, as follows:

// Flush the TCP/IP Protocol

Syslog.getInstance("tcp").flush();

**2.11. How do I properly shut down all client protocols?**

Applications which use only UDP/IP-based, Unix socket, and/or Unix

syslog client protocols do not need to be explicitly closed.

When shutting down a servlet-based application, it's a good idea to

call Syslog.shutdown() in the servlet's destroy() method. Calling

Syslog.shutdown() ensures that all client connections close properly.

When calling Syslog4j from a standalone Java application, it's a good

idea to both call a small Thread.sleep(..), say 500 milliseconds or so,

and then call Syslog.shutdown(). JUnit tests can also benefit from

this sort of paused shutdown.

**2.12. How do I run a command-line syslog client with Syslog4j?**

To access the syslog client, run this (substituting the appropriate

version for x.y.z):

java -cp syslog4j-x.y.z.jar org.productivity.java.syslog4j.SyslogMain

...or (alternately) this shortcut technique:

java -jar syslog4j-x.y.z.jar

Running one of the above commands will provide a list of command-line

options. Here's an example which sends a UDP/IP test message to a host

located at 192.168.0.100 on port 5555:

java -jar syslog4j-x.y.z.jar -h 192.168.0.100 -p 5555 udp "test"

Only the stock protocols are currently supported in the Syslog4j command-

line client.

**2.13. How do I run a command-line syslog server with Syslog4j?**

Running a command-line syslog server is extremely handy when trouble-

shooting your application. Run this to get a list of command-line

options (type all of this on one line):

java -cp syslog4j-x.y.z.jar

org.productivity.java.syslog4j.server.SyslogServerMain

Here's an example which waits for requests on port 5555 and displays

the output to standard out:

java -cp syslog4j-x.y.z.jar

org.productivity.java.syslog4j.server.SyslogServerMain -p 5555

To test it, open another window/shell and run the following:

java -jar syslog4j-x.y.z.jar -p 5555 udp "test"

...which should show the message in the server window/shell.

The command-line Syslog server can also write to a file using the -o

option. However, it's recommended to not use the command-line

syslog server for production purposes; you're best to look into

solutions such as syslog-ng from Balabit IT Security for a robust

syslog server implementation.

Only the "udp" and "tcp" protocols are currently supported in the Syslog4j

command-line server.

--------------------------------------------------------------------------

**3. Syslog4j and Apache Log4j (http://logging.apache.org/log4j/)**

**3.1. Is Syslog4j meant to replace Log4j?**

Absolutely not! Log4j is a robust and complete logging solution.

Syslog4j is meant as a low-level implementation of the syslog

protocol, and not designed to serve as a replacement to Log4j.

**3.2. Wait, doesn't Log4j come with a syslog appender?**

In fact, it does! If an application already uses Log4j, the appender

class of org.apache.log4j.net.SyslogAppender may suit the

majority of UDP-based syslog messaging and Syslog4j isn't needed.

**3.3. So why would I bother using Syslog4j if I already use Log4j?**

Syslog4j provides support for protocols other than UDP, which is

the only network protocol supported by the Log4j's SyslogAppender.

The SyslogAppender provided by Log4j is hard-coded to use the

standard syslog port 514 with a maximum message size of 1024

bytes and currently does not support Structured Syslog messages

as per RFC 5424.

**3.4. How do I use Syslog4j with Log4j?**

Syslog4j provides a Log4j appender called "Syslog4jAppender," not

to be confused with "SyslogAppender," which comes stock with Log4j.

The Syslog4jAppender allows you to use all Syslog4j protocols

within Log4j. This includes stock and custom protocols.

**3.5. How do I configure Log4j to use the Syslog4jAppender?**

Here is an example log4j.xml entry, which uses the Syslog4j stock "udp"

protocol:

<appender name="Syslog4j" class="org.productivity.java.syslog4j.impl.log4j.Syslog4jAppender">

<param name="Protocol" value="udp"/>

<param name="Facility" value="user"/>

<param name="SyslogHost" value="192.168.0.100"/>

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="%d{ABSOLUTE} %-5p [%c{1}] %m"/>

</layout>

</appender>

To use a custom Syslog4j configuration, simply create your own Syslog4j

protocol with Syslog.createInstance(protocol,config) and configure the

log4j.xml "Protocol" parameter as your chosen "protocol" name.

--------------------------------------------------------------------------

**4. PCI DSS Audit Logging with Syslog4j**

**4.1. What is PCI DSS?**

From:

https://www.pcisecuritystandards.org/security\_standards/pci\_dss.shtml

"The PCI DSS, a set of comprehensive requirements for enhancing payment

account data security, was developed by the founding payment brands of

the PCI Security Standards Council, including American Express,

DiscoverFinancial Services, JCB International, MasterCard Worldwide and

Visa Inc. Inc. International, to help facilitate the broad adoption of

consistent data security measures on a global basis.

The PCI DSS is a multifaceted security standard that includes

requirements for security management, policies, procedures, network

architecture, software design and other critical protective measures.

This comprehensive standard is intended to help organizations proactively

protect customer account data."

**4.2. What is PCI DSS Audit Logging?**

Section 10.3 of the PCI DSS specification (pages 46-47 of version 1.2,

published in October 2008) requires the following fields be logged

for audit purposes:

10.3.1 User Identification

10.3.2 Type of event

10.3.3 Date and time

10.3.4 Success or failure indication

10.3.5 Origination of event

10.3.6 Identity or name of affected data, system component, or resource

**4.3. How does Syslog4j facilitate PCI DSS Audit Logging?**

Syslog4j provides a Java object named PCISyslogMessage which conveniently

handles these fields.

**4.4. How do I use PCISyslogMessage?**

PCISyslogMessage has several constructors which cover some or all of the

fields in PCI DSS section 10.3. An example using the appId, userId,

eventType, status, and affectedId fields is shown below:

// Create a PCISyslogMessage

PCISyslogMessage message =

new PCISyslogMessage("myapp","jsmith","login","success","jsmith");

// Log the PCISyslogMessage

Syslog.getInstance("udp").info(message);

PCISyslogMessage can also be constructed with (1) a Map object with

key/value pairs containing the above fields, or (2) an object

implementing the PCISyslogMessageIF interface, or (3) the default

constructor. PCISyslogMessage provides standard "Javabean" getters

and setters for all PCI DSS audit fields.

--------------------------------------------------------------------------

**5. Miscellaneous**

**5.1. Who designed the Syslog4j logo?**

Tom Spence designed the excellent Syslog4j logo based on a written

description of the project. Thanks, Tom!

You can contact Tom and see his other art at: http://growtreeisland.com

--------------------------------------------------------------------------