Fusion Syslog Server User Manual

2013R1

Table of Contents

1 Document Introduction

1.1 Document Purpose

The purpose of the document is to explain how Fusion Syslog Server works and how to configure it. Installation is covered in [1].

1.2 Document Audience

The readers will be Fusion Administrators.

1.3 Document History

Version	Editor	Date	Changes
1.2.0	M. Simonsen	23-Feb-09	Initial public version.
1.2.1	M. Simonsen	23-Mar-09	Revised version
1.2.2	M. Simonsen	03-Apr-09	Revised edition
1.2.3	M. Simonsen	26-Jun-09	Revised edition
1.2.4	M. Simonsen	16-Nov-09	Revised edition
1.2.7	M. Simonsen	27-Sep-10	Revised edition
1.2.8	J. Hübenthal	19-Okt-10	Revised edition
1.2.9	M. Simonsen	05-Jan-11	Revised edition
1.2.10	M. Simonsen	07-Feb-11	Bugfix edition for 2011R1-SP1
1.3.5	M. Simonsen	19-Dec-11	2012R1 - changed name
1.4.18	M. Simonsen	20-Feb-13	2013R1 – small changes

1.4 References

Document
[1] Fusion Installation

2 Introduction

Fusion Syslog Server is a standard syslog server, fully compliant with RFC3164. The value of such a server is very high if you have devices that are capable of logging syslog messages. In that case you can redirect your devices to log to Fusion Syslog Server, and start to take advantage of all the information the devices can give you.

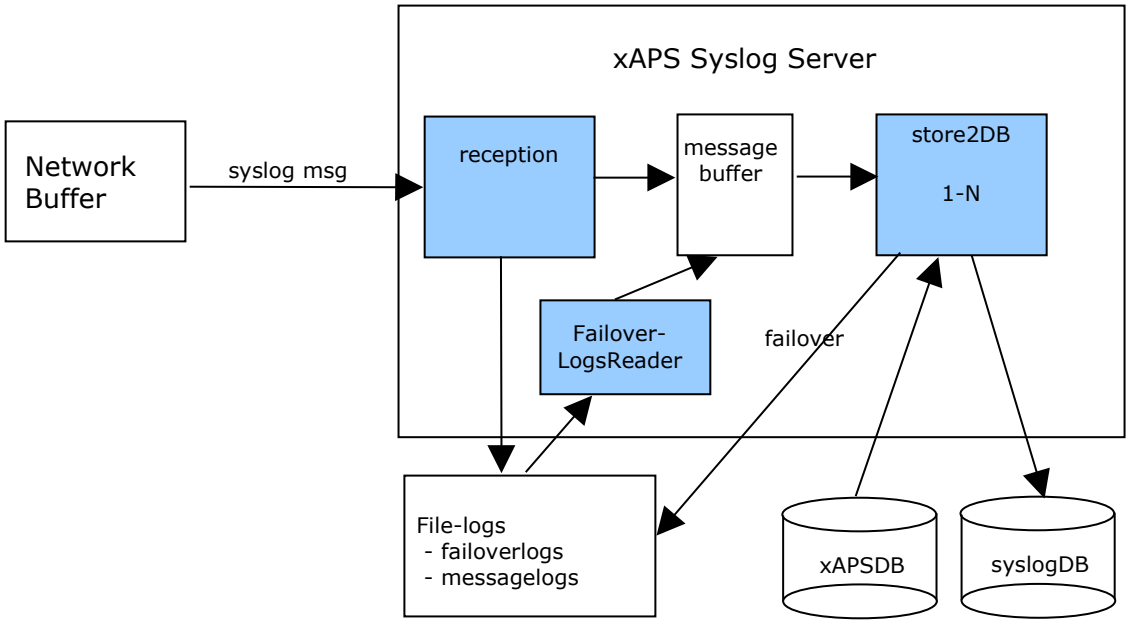
NB!! Is is required that a Fusion Syslog Server is always present at every host running a Fusion module. Even if you decide to run your syslog database (and also the syslog server) on another hose, you still must setup the syslog server on all hosts running a Fusion module (web, shell, etc). In case of such a setup, the server must be configured to run as a relay-server – check the configuration!

Fusion Syslog Server receives the messages and parses them. Then it store the messages to file and/or to the Fusion Syslog DB. The server is build to be able to handle large bursts of syslog messages from the devices, so there shouldn't be any performance issues. You can also control some of the behaviour of the server through configuration.

As of today, it should not be a problem to handle at least 10K messages pr minute. However, such large amount of data could take time to sift through, so it may require a very fast disk plus a healthy amount of memory.

3 Design and message flow

A goal for this design was to make a high performance server. At the same time we wanted to protect the SyslogDB against an enormous burst of activity from the devices. Such a burst could be when all devices come on-line after a power outage or that many devices are set in debug-mode. To get both performance and protection of the database, we rely on a message buffer which caches the messages for a while until the database storage threads are finished. Let's look on a drawing of this message flow:



The overall picture is that a syslog message is picked up by the server and written to syslog database and possibly some file logs. The blue coloured boxes represents separate threads/processes inside the server, so each of these boxes operate independently of each other. The white boxes represent buffers/files/databases – or in general: storage places. A buffer is an in-memory storage to stuff syslog messages.

The reception process is designed to capture syslog messages and dump them into the message buffer as fast as possible. After the message is written to the message buffer, the message may be logged to a file (messagelogs) if so decided by the configuration. When the message buffer is full, the reception thread will start to write message to failover logs on disk. If reception works too slow network buffer will probably overflow with syslog messages, and then some messages will be lost (since syslog is using UDP). The only reason why reception would be too slow is if the file writing is going slowly, so it's a good idea to have fast disk access on a syslog server.

The message buffer is a FIFO (First In First Out) queue in memory. The size of the buffer is configurable and set to 100 000 as a default value. According to the syslog specification, no syslog message can be more than 1024 byte[1], but since the messages is

[1] The server will accept any length up to the standard network buffer size in the OS. On Windows Vista 32-bit that is 8192 bytes. However, the database will not accept messages longer than 1024 bytes, so

stored as Java Strings we can estimate each syslog message to approximately 2 KB in average. Then we can compute the total amount of memory needed for 100 000 messages: 100K * 2KB = 200MB. If the message buffer reaches its limit, then the messages are written to file (failoverlogs). If the server crashes, all messages in the buffer are lost, if not messagelogs are written.

The store2DB process is multithreaded, to overcome slow network access to the syslog database. You can configure how many instances of this process you want to run. If you decide to let many threads run then you can expect fewer messages in buffer and faster insertion to the database. However, the flip side of this is that the syslog and xaps database may be too heavy loaded, since bursts of messages will instantly be felt on the database side of the syslog server. We recommend to set the number of store2DB threads to 1 if, and only if, this is fast enough to get all messages in within a couple of minutes and maybe a little more in the event of large message "storms".

If something goes wrong in the process of storing the message to the database, the message is written to failover log. The arrow pointing from Fusion DB is to show that we need information about unittype and profile from the Fusion DB before we store all the syslog information + extras into the syslog database. So for each message into Fusion Syslog Server we need to do one query on Fusion database and one insert into syslog database.

Since the Fusion database is the same database the provisioning servers depend on to do their job, we have made it possible to setup the syslog database on a separate physical machine. For this reason you can configure two database connections in the configuration. Although not recommended you can turn off the Fusion DB check (to increase performance even more), but then you will not get information about unittype/profile in the syslog database.

The FailoverLogsReader is responsible for reading the failover log and putting in back into the buffer. The process interval for the failover logs can be configured, and it makes sense to not do this too often. The reason for this is that the situation that initially caused failover messages to be written might not disappear very quickly (it could be database downtime or a burst of messages). It may of course be that the same error occurs again so that new failover logs are written each time the failover log is processed. For that reason store2DB will check the timestamp in the failover log and discard the message if it is older than 24 hours (this is also configurable).

To further enhance the utilization of resources on the system, the syslog server has some mechanisms to shift priorities in the event of a burst of messages. In this event, the store2DB threads will run at a slower pace, allowing the system to use as much resources as possible to retrieve all messages and store them either in the message buffer or on failover logs. As soon as the message buffer starts to decrease, the syslog2DB threads resume normal operation at full speed. Likewise that failover log reader (which feed back syslog message into the system) will halt if there is a situation of "high receive load".

Tests shows that the server should handle at least 30K messages pr minute (sustained load) and much higher load in shorter periods of time. Initial receive capacity is at least 1M messages pr minute.

---

that is really the upper limit. Long messages will then be truncated to 1024 bytes content.

4 Configuration

4.1 xaps-syslog.properties

```
# --- Syslog Server ---

# Port to listen to. Default is 9116
port = 9116

# This buffer contains the UDP messages received by the OS, not yet
# processed or accepted by the syslog server. This buffer is important if
# many messages arrive at the same time. Specify in KB. Default is 10240.
receive-buffer-size = 10240

# This buffer contains the syslog-messages received. Then syslogdb-threads
# read from this buffer and store it to the database. This ensures that
# the syslog server can receive large quantities of messages, without putting
# an enormous load on the syslogdb (which is of utmost important). You
# decide how many messages this buffer can contain. Remember to up your
# memory-settings on this server if you set it to a high number. Default
# is 100000. (Should amount to approx. 2KB*100K = 200MB memory depending
# on the length of the messages).
max-messages-in-buffer = 10000

# When this limit is reached, the server will pause. Operations will resume
# as soon as disk space is freed. However, already at twice this limit, the
# server will stop accepting new syslog messages, and only process already
# accepted messages (which may be found in message-buffer or in failover-log
# files). Specify in MB. Default is 100.
min-free-disk-space = 100

# Decide how many threads/processes will run simultaneously and store syslog
# messages to database. Default is 1.
max-syslogdb-threads = 1

# Decide how old a failover message can be before we don't retry the message.
# Default is 24. Specify in hours.
max-failover-message-age = 24

# The interval failover-logs are processed and fed back into the system.
# Consider these two cases were failover is active:
# - burst of syslog messages, causing the message-buffer to overflow
# - temporarily error on the database
# Both of these cases might be fixed within a certain amount of time. It
# then makes no sense to try to restore failover message more often than
# the minimum time expected to fix one of these issues (burst are over or
# datatbase up again). Specify in minutes. Default is 30.
failover-process-interval = 30


# Each message should contain a unique key to connect it a Unit. This
# key is usually a MAC address or maybe a serial number. As long as this
# MAC or serial number is a parameter found in the Unit, then Fusion
# can "connect" the syslog data to the Unit. This "connection" is the
# goal of this piece of configuration. To identify this unique key
# we have a default pattern (a regular expression):
#
#    '\\[([a-fA-F0-9:-]{12,17})\\]:'
#
# The parenthesis marks the unique key, which in this case is a MAC
# address. The usage of regular expression is pretty complex, but the
# translation of the pattern above goes like this:
#
# The pattern must start with '['. However, '[' is a special char
# in regex, so we need to escape it with a double \\. The parenthesis
# marks a regular expression group. The unique key is found within
# this "group". '[a-fA-F0-9:-]' is to say that the pattern will match
```

```
# with any letter from a to f (lower or upper case), any number, colon
# or hyphen. '{12,17}' says that the previous pattern must be repeated
# from 12 to 17 times. Lastly a ']:' must follow, but is not part
# of the unique key.
#
# The end result of this pattern is that the following strings will match:
#
# [123456789012]:
# [aa-bb-cc-dd-ee-ff]:
# [00:11:22:aa:FF02]:
#
# However, your syslog message may not match this pattern. Therefore
# Fusion Syslog server offers the ability to add your own patterns by
# adding properties in this manner:
# deviceid.pattern.1 = pattern1
# deviceid.pattern.2 = pattern2




# Some devices might connect to the syslog server even if they're not registered
# in xAPS. Three options are possible:
# 1. Allow the into the syslog anyway (default).
# 2. Allow and register a dummy-unit into a default MonitorDevice unit type
# 3. Discard
# 4. Redirect to another host
# In case you decide to host the syslog database on a separate server, you still
# must run a syslog server locally on every server which hosts a Fusion module.This
# "local" syslog server will then act as mere relay, and consequentially must
# redirect to the remote syslog server using option 4.
# Formal option description:
# allow | allow-create | discard | host[:port]
unknown-units = discard

# The server will keep track of duplicate messages and discard duplicates according
# to the task in the matching syslog-event. For this matching of duplicates the
# server maintains a buffer which can grow quite large. Duplicate matching will
# only be initiated for those messages with a DUPLICATE syslog-event.
# Set the maximum size here. Specify in number of messages. Default is 100000.
max-message-in-duplicate-buffer = 100000

# --- Syslog Database ---

# Format: name/password@database-url
db.syslog.url = xaps/xaps@jdbc:mysql://localhost:3306/xaps

# --- Fusion/xAPS Database

# If you want to keep the xAPS database on a separate server from the syslog
# database you can specify the url here (see format above). The syslog server
# needs to lookup unit/profile/unittype information from xAPS. In order to minimize
# the impact on xAPS from syslog it can be smart to move the syslog database to a
# separate server. It is possible to set db.xaps = db.syslog, if the databases are
# on the same server.
db.xaps = db.syslog
```

Decide port number (default syslog server port is 9116) and set your devices to log to this port number.

Decide the receive buffer size. This buffer size doesn't need to be as big as the other buffers, since the process of retrieving from receive buffer and place into the message buffer is very fast. So compared to messages-in-buffer (approx 2KB pr message), there should probably be a 1:10 relationship. E.g.: receive-buffer = 20480 (20MB) and max-messages-in-buffer = 100000 (200MB). On Linux, /proc/sys/net/core/rmem_max defines the maximum receive buffer size, usually set to 128KB; sysctl is the appropriate tool to change this.

Nothing is more annoying than having a break down in the service because of disk space. This server can probably fill up your disk space (with failover logs) if the sustained syslog message input is too high or the database is not functioning properly. To avoid a situation where this server makes a problem for other services, define a lower limit of the free disk size. At twice this limit the server will stop accepting new syslog messages, and as the free disk size decreases beyond the limit, the syslog2DB threads will also stop to write to the database.

Next, decide the number of threads to store to syslog database. Set as low as possible (meaning "possible" in the sense that it is enough to process the average number of messages coming in to the syslog server) because it distribute the load on syslog and Fusion database more evenly.

Then decide how old a failover message can be before we discard the message. Don't set the value too high because if a message fails time and time again it will cause unnecessary load on the system.

Failover-process-interval should be set to 15-60 minutes. Setting it to a very low value will cause more stress on your system. However, to prevent unnecessary stress, the failover log feedback (which reads from the failover log and inserts into the message buffer) will halt as soon as the server detects that something is writing to failover log.

The next property to decide is not easy. This is because each device might send the unit id identifier in different ways inside the so-called CONTENT of the syslog message. In Ping Communication we have suggested that this unit id should be written like this: [<MAC-address>]: with the rest of the syslog message content trailing. If you check with RFC3164 you will see that we have replaced the place where one traditionally has a pid with the device's MAC address. To specify how to find the unit id we require a regular expression. Regular expression is a language in itself and you cannot change this property without studying this format. If you know the format then you just need to remember to enclose the unit-id pattern with parentheses.

Next decide whether you will accept syslog messages not known in Fusion. The problem with allowing such messages is that you could get messages which have nothing to do with Fusion at all. On the other side you can increase performance by not asking Fusion for information about profile/unittype/verification. A third option is to redirect the messages to another syslog server. This is not truly a relay function as specified in RFC3164, but it works anyway!

Duplicate messages can or cannot be something of interest. It is possible to discard duplicate messages in the syslog server, to avoid a lot of unnecessary stuff in the database. The duplicate timeout rule is specified in the syslog event corresponding to the syslog message. However, such a duplicate matcher system will need some amount of memory. Set the size of the buffer, which will then contain one of each type of message. The duplication must match both on unitId and content of the message. If the buffer size is too small, then duplicate messages will be written to database.

The database configuration is specified on one-line format:

<user>/<password>@<jdbc-database-url>

syslog database must be specified, but Fusion may be set to "db.syslog". That is the same as saying these two database are found in the same database. To identify the

correct jdbc-database-url, check database documentation, but both MySQL and Oracle format is shown here.

The last configuration to do is to decide how long the syslog server should cache certain key information from the xaps database. This object is about profile names and unit type names and is only used when and old style Owera syslog messages are used, otherwise is this information is retrieved directly in the query to Fusion.