opEvents - Centralized Logging Solution

Generally speaking, the overall goal is to provide centralised logging services for the purposes of operations, compliance and audit. All systems should send log messages to the central server, where the logs will be kept in original form for the required period of time and the logs will be available for event, incident and problem management purposes.

Centralised Logging Architecture

Architectural Considerations

There are several considerations for creating a centralised logging solution.

- What devices and operating systems will be sending logs.
- What applications will be sending logs.
- What protocol will be used to send log messages.
- What timezone are each of the devices sending logs in.
- What criticality of logs is required.

Devices Sending Logs

You should complete a table as below to catalogue all the devices which will be sending logs.

Name	Purpose	Logging System
Windows Event Manager	Event and Audit	Windows Event Logging
CentOS Linux 5.x	Event and Audit	syslog
CentOS Linux 6.x	Event and Audit	syslog
Cisco IOS Switches	Event and Audit	Cisco IOS syslog
Cisco IOS Routers	Event and Audit	Cisco IOS syslog

Applications Sending Logs

The following applications logs need to be send centrally.

Application Name	Purpose	File	Device
Monkey Auth System	Audit	C:\Program Files\MAS\logs\monkeyauth.log	Windows 2008 Servers
Elephant Financial	Audit	/data/elefin/log/app.log	CentOS Linux 6.8

Logging Protocol

syslog has proven to be a very robust protocol for large scale log management.

TCP should where reliable logging is required, UDP works very well, 99.99% of the time.

Logging Severity Levels

The requirement is to send level 6 and above.

Value	Severity	Keyword
0	Emergency	emerg
1	Alert	alert
2	Critical	crit
3	Error	err
4	Warning	warn
5	Notice	notice
6	Informational	info
7	Debug	debug

Full details for syslog severity levels https://en.wikipedia.org/wiki/Syslog#Severity_level

Handy cross reference: opEvents priority levels vs. NMIS and Syslog levels

Time Handling

There are two primary concerns about time, one clock drift the second is multiple timezones.

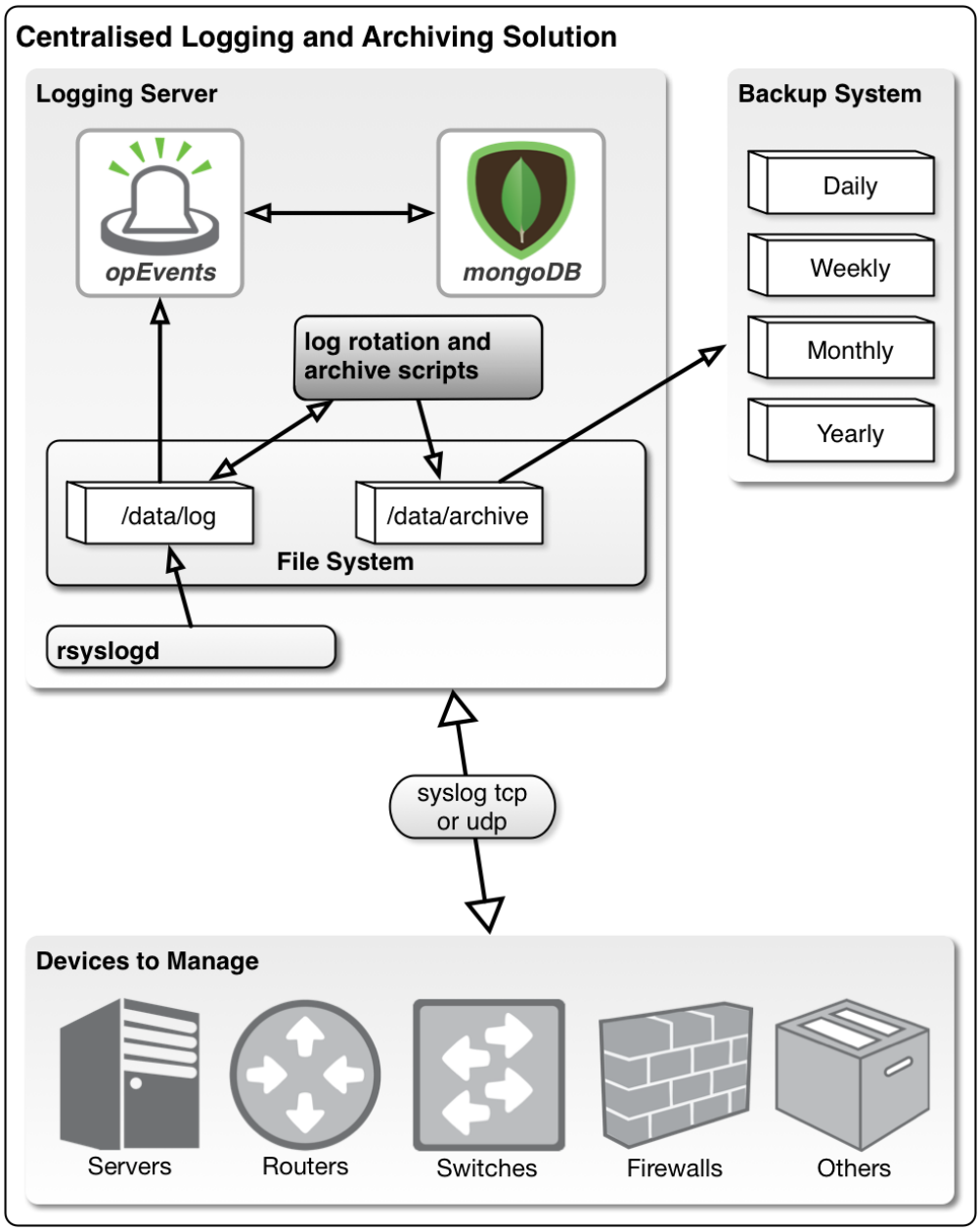
There are several options for handling timezone, the following are considered best practices for IT management in this regard:

- All devices should use NTP to ensure accurate time is set
- All devices should have their timezone set to their localtime or to a common timezone, e.g. UTC
- All devices should include the timezone when timestamping, ideally as an offset, e.g. +00

However it is often difficult to get all devices especially devices already installed to do all these things, so a great option is to make the logging server the authoritive time, NTP is setup a timezone is selected, and all logs received by it will be stamped with the time when the log is received.

Centralised Logging and Archiving Solution

The following diagram shows how a system can be implemented using syslog and Opmantek opEvents.



Centralised Logging Design

Syslog Method and Transport

The following table summarises how the logs will get from the source into the opEvents server. Three basic techniques exist, native syslog transport supported by the operating system, a logging agent which monitors for data and sends the resulting data/events/log as a syslog.

There are several good choices for Windows, but NXLOG has proven to meet all the requirements, almost all other systems include embedded syslog systems

Source	Method and Transport
Windows 2003 Servers	nxlog monitoring Windows Event log, transport over syslog
Windows 2008 Servers	nxlog monitoring Windows Event log, transport over syslog
Windows 2012 Servers	nxlog monitoring Windows Event log, transport over syslog
CentOS Linux 5.x	rsyslog 3.x

CentOS Linux 6.x	rsyslog 7.6
Cisco IOS Switches	Native IOS syslog
Cisco IOS Routers	Native IOS syslog
Monkey Auth System	nxlog running on Windows.
Elephant Financials	rsyslog running on Linux

syslog Facility

The best reference is: https://en.wikipedia.org/wiki/Syslog#Facility

We are primarily concerned with the facilities localX facilities. Logs will also grow at different rates and having them in separate files will allow for more granular control. The following table summarises which log files will end up in which files.

Device Type	syslog facility	Log file
	local0	/data/log/local0.log
Log server to log server (future)	local1	/data/log/local1.log
Application logging e.g. MonkeyAuth	local2	/data/log/local2.log
Windows servers (nxlog default)	local3	/data/log/local3.log
Cisco ASA default (VMware ESXi default)	local4	/data/log/local4.log
	local5	/data/log/local5.log
Linux syslog	local6	/data/log/local6.log
Cisco Routers and Switches	local7	/data/log/local7.log

Alternate file naming can be supported if required, e.g. cisco.log instead of local7.log.

Centralised Logging Implementation

Central rsyslog Server Configuration

Translating all the above into the configuration the following are the most important parts.

It should be noted that trying to use the /etc/rsyslog.d scheme did not work.

Enable rsyslog to receive UDP and TCP syslogs

By default (to prevent DOS) rsyslog is configured to not receive syslogs from remote servers.

```
# Provides UDP syslog reception
$ModLoad imudp.so
$UDPServerRun 514

# Provides TCP syslog reception
$ModLoad imtcp.so
$InputTCPServerRun 514
```

Configure each Facility to be saved into Files

Based on the table above the following would be the configuration

```
local0.*        /usr/local/nmis8/logs/local0.log
local1.*        /usr/local/nmis8/logs/local1.log
local2.*        /usr/local/nmis8/logs/local2.log
local3.*        /usr/local/nmis8/logs/local3.log
local4.*        /usr/local/nmis8/logs/local4.log
local5.*        /usr/local/nmis8/logs/local5.log
local6.*        /usr/local/nmis8/logs/local6.log
local7.*        /usr/local/nmis8/logs/local7.log
```

Configure syslog reception for remote Linux hosts

Based on the table above the following would be the configuration

```
$template LinuxLogs,"%timegenerated%.%timegenerated:::date-subseconds% %HOSTNAME% %syslogtag%%msg%\n"
if      $fromhost-ip != '127.0.0.1' \
        and $syslogseverity <= '6' \
        and $syslogfacility <= '15' \
then    /usr/local/nmis8/logs/linux.log;LinuxLogs
```

Optionally handling things with the WORD LINUX in the tag

```
if      $fromhost-ip != '127.0.0.1' \
        and $syslogtag contains 'LINUX' \
        and $syslogseverity <= '6' \
        and $syslogfacility <= '15' \
then    /usr/local/nmis8/logs/linux.log;LinuxLogs
```

Handling Different Times and Time Zones

The following configuration shows how to create an rsyslog template and apply that to the logs being received, this example also adds high precision time, which is supported by opEvents.

```
$template ServerTime,"%timegenerated%.%timegenerated:::date-subseconds% %HOSTNAME% %syslogtag%%msg%\n"
local5.*        /usr/local/nmis8/logs/local5.log;ServerTime
```

All syslog received to the facility local5 will be timestamped with the receiving syslog servers high precision time.

Sample Configuration for rsyslog 7.6

The following is a config which sends logs from /var/log/messages using facility local6.

```
# enable the imfile module for file monitoring
$ModLoad imfile
$WorkDirectory /var/spool/rsyslog

# Monitor the file
$InputFileName /var/log/messages
$InputFileTag :
$InputFileStateFile messages_log
$InputFileSeverity error
$InputFileFacility local6
$InputFilePollInterval 1
$InputFilePersistStateInterval 1
$InputRunFileMonitor

# forward these logs to another server
local6.*                                @192.168.1.7:514
```

Remote Linux Server

The following rsyslog config will send all syslogs which are sourced localy with a severity 0-6 to the remote server.

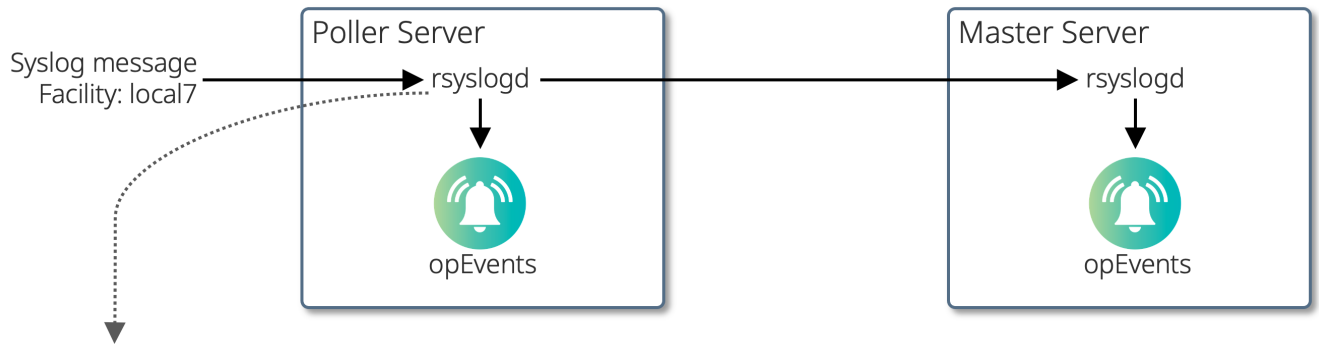
```
$template LinuxMnemonic,"%timereported% %HOSTNAME% %syslogfacility-text%-%syslogseverity%-%syslogtag%%msg%\n"
if $fromhost-ip == '127.0.0.1' and $syslogseverity <= '6' then @192.168.1.7;LinuxMnemonic
```

Here we are using a Linux Mnemonic like the Cisco Syslog so that we do not lose the original facility and severity when the message is fowarded.

Optionally send everything with the WORD LINUX in the tag

```
$template LinuxMnemonic,"%timereported% %HOSTNAME% LINUX-%syslogfacility-text%-%syslogseverity%-%syslogtag%%msg%
\n"
```

Example Topology




```
### /etc/rsyslog.d/nmis.conf

# and handle inbound/poller NMIS syslogs
local7.*   /usr/local/nmis8/logs/cisco.log
local7.*.  @10.215.1.5:514
local1.*   /usr/local/nmis8/logs/poller_event.log
```

In the example above all syslog messages received with a facility of local7 will be forwarded to the Primary server at 10.215.1.5. When this message is forwarded from the poller to the Primary, the poller will insert its own timestamp into the message.

```
### syslog message from the node to the poller server

02:23:37.250516 IP 10.10.1.1.58415 > 10.215.1.7.syslog: SYSLOG local7.notice, length: 100
E....Y.....+
..
..../...l./<189>90: *Feb  1 11:23:35.623: %SYS-5-CONFIG_I: Configured from console by hero on vty0 (10.215.1.5)
^C

### syslog message from the poller server to the primary server

11:23:37.273514 IP 10.215.1.7.35902 > 10.215.1.5.syslog: SYSLOG local7.notice, length: 126
E.....@.?.#.
...
....>.....j<189>Feb  1 02:23:37 10.10.1.1 90: *Feb  1 11:23:35.623: %SYS-5-CONFIG_I: Configured from console by
hero on vty0 (10.215.1.5)
```

If the servers/nodes are in differnt time zones or the clocks are not correct opEvents may not by default accept the syslog message. Setting the opEvents debug option to '1' will show the following message in /usr/local/omk/log/opEvents.log if this is the case.

```
[Wed Feb  1 09:08:49 2017] [debug] worker[4697] event 1485875324 R1 Feb  1 00:08:44 10.10.1.1 66: *Feb  1 09:08:
42.711: %OSPF-5-ADJCHG: Process 1, Nbr 10.10.1.3 on FastEthernet1/0 from FULL to DOWN, Neighbor Down: Interface
down or detached is older than opevents_max_event_age, skipping!
```

As of opEvents 2.2.1 we can provision opEvents to allow these wildly out of date syslog messages and replace the origin timestamp with its own.

```
### /usr/local/omk/config/opCommon.nmis

    'opevents_max_action_queue_age' => 3600,
    'opevents_max_event_age' => 7200,
    'opevents_max_time_delta' => 1800,
    'opevents_monthly_report_recipients' => [],
    'opevents_monthly_report_title' => 'Monthly Summary Report',
```

Setting the 'opevents_max_time_delta' to a value of 1800 seconds will instruct opEvents to do the following:

- If the syslog message has a timestamp that is more than 1800 seconds off from the current server time:
    - Accept the syslog message
    - Remove and replace the timestamp with its own time stamp.

References

https://en.wikipedia.org/wiki/Syslog#Facility

http://wiki.rsyslog.com/index.php/Sysklogd_drop-in_with_remote_logs_separated_by_dynamic_directory

http://man7.org/linux/man-pages/man5/rsyslog.conf.5.html

http://www.rsyslog.com/doc/v7-stable/configuration/property_replacer.html

http://www.rsyslog.com/doc/v7-stable/concepts/multi_ruleset.html

http://fibrevillage.com/sysadmin/221-rsyslog-rules-examples-on-linux

http://superuser.com/questions/532587/configure-rsyslog-server-to-log-incomming-messages-with-time-of-the-rsyslog-serv

http://people.redhat.com/pvrabec/rpms/rsyslog/rsyslog-example.conf

Appendix A: Upgrading rsyslog of RHEL and CentOS

The version of rsyslog which is included in older CentOS and RedHat systems is problematic and should be upgraded, the steps to do so are included below.

Create/Edit rsyslog.repo

```
cat /etc/yum.repos.d/rsyslog.repo
```

Result:

```
[rsyslog_v7]
name=Adiscon CentOS-$releasever - local packages for $basearch
baseurl=http://rpms.adiscon.com/v7-stable/epel-$releasever/$basearch
enabled=1
gpgcheck=0
gpgkey=http://rpms.adiscon.com/RPM-GPG-KEY-Adiscon
protect=1
```

Test Yum

```
yum search rsyslog
```

Install rsyslog

```
yum install rsyslog
```

Related Topics

- opEvents - Syslog Handling - Adding a New Vendor
- opEvents - Syslog Handling - Adding a New Format
- SNMP Traps with Cisco and Other devices
- High Volume SNMP Trap Processing