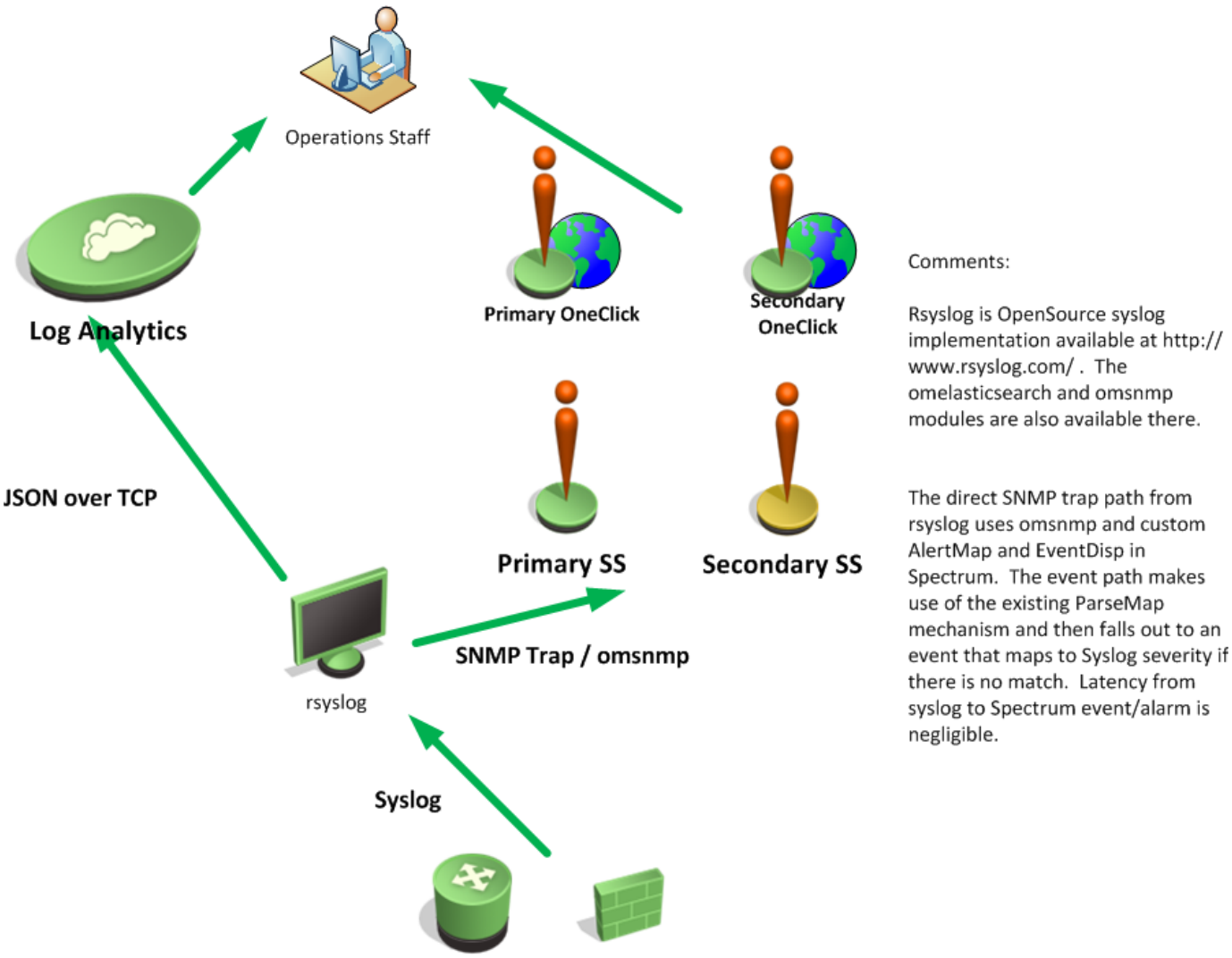
RSyslog Integration

For many years, Spectrum has supported ingestion of Syslog sourced events via log match traps from iAgent (SNMP Research CIAgent), CA SystemEDGE, and CA NSM. Given the current state of these agent technologies, there is a need for additional methods to send Syslog data into Spectrum, particularly since it is an important source of network events, in addition to SNMP traps and polling. Rsyslog is a popular Syslog server that is also used with CA Digital Operational Intelligence and Log Analytics for CA UIM. One of the interesting features of Rsyslog is the ability to forward messages as SNMP traps via the omsnmp module.

What this allows you to do is configure something like the following:



If you're using an RPM based Linux distribution (Red Hat, CentOS, etc.), the two required packages are rsyslog and rsyslog-snmp.

Once those packages are loaded, you'll need to configure RSyslog to receive Syslog messages from network devices and forward them as traps to Spectrum. The configuration file you'll need to edit is /etc/rsyslog.conf. Here are the relevant parts:

Input Modules

Rsyslog has the concept of input modules that allows Rsyslog to consume data from log files, UNIX sockets, TCP and UDP, and other sources. The typical Rsyslog configuration will have modules for UNIX socket, Kernel log, and perhaps Systemd Journal but for receiving Syslog data from network devices, the imudp module will be required:

```
#### MODULES ####
module(load="imuxsock") # provides support for local system logging (e.g. via logger command)
module(load="imklog")   # provides kernel logging support (previously done by rklogd)
#module(load"immark")  # provides --MARK-- message capability
# Provides UDP syslog reception
# for parameters see http://www.rsyslog.com/doc/imudp.html
module(load="imudp") # needs to be done just once
```

In addition to loading the module, we can specify parameters for operation. At minimum, the listening port number needs to be specified. In this configuration, we're also going to specify a ruleset called "network" for use with the UDP input module:

```
#input(type="imudp" port="514")
input(type="imudp" port="514" ruleset="network")
# Provides TCP syslog reception
# for parameters see http://www.rsyslog.com/doc/imtcp.html
#module(load="imtcp") # needs to be done just once
#input(type="imtcp" port="514")
```

Specifying a ruleset provides the opportunity to control how the incoming messages are to be processed, formatted, and ultimately forwarded. There is also a TCP input module and if you have devices that use TCP to send their Syslog data, you'll want to enable that and specify the "network" ruleset like was done for UDP.

Output Modules

Similar to input modules, output modules allow Rsyslog to forward processed log data to log files on disk, TCP or UDP, databases, as SNMP traps, etc. The omfile (writing to disk) and omfwd (forward via TCP or UDP to something like Log Analytics) modules are built in but omsnmp (SN MP trap) needs to be specified to load:

module(load="omsnmp")

Templates

Rsyslog makes use of templates to specify how forwarded/saved messages should be formatted. If Log Analytics is part of the configuration, there will be a template called "ls_json" that is outlined in the appropriate documentation (step 3). For SNMP traps to Spectrum, we can use the following:

```
template(name="SNMPFormat" type="list") {
    constant(value=" TS:")
    property(name="timestamp" dateFormat="rfc3339")
    constant(value="SFP=")
    property(name="syslogfacility")
    constant(value=":")
    property(name="syslogpriority")
    constant(value=" Src: ")
    property(name="fromhost-ip")
    constant(value=" Tag=")
    property(name="syslogtag" position.from="1" position.to="32")
    constant(value=" Msg: ")
    property(name="msg" spifno1stsp="on")
    property(name="msg")
    constant(value="\n")
    }
```

The template above would result in a trap with the significant varbind that looks like this:

```
TS:2018-05-09T17:30:46.503221-04:00SFP=23:4 Src: 192.168.75.140 Tag=49: Msg: *May 9 21:30:45.493:
%IOSXE-4-PLATFORM: R0/0: kernel: hrtimer: interrupt took 2603406 ns
```

In the template, there are references to properties. If needed, properties can be substituted. For example, the template provided uses "fromhost-ip" but if the syslog messages aren't coming directly from the managed devices, you'll want to use "hostname" for the property instead and make sure that name resolution is working properly in the environment. Additionally, there are properties for facility and severity (listed as syslogpriority). The severity can be used for determining the severity (critical, major, minor, etc.) of the alarms in Spectrum. If you would like to change the severity of the alarm in Spectrum from what the origin believes it should be, you could have additional templates:

```
template(name="SNMPFormat_critical" type="list") {
    constant(value=" TS:")
    property(name="timestamp" dateFormat="rfc3339")
    constant(value="SFP=")
    property(name="syslogfacility")
    constant(value=":2 Src: ")
    property(name="fromhost-ip")
    constant(value=" Tag=")
    property(name="syslogtag" position.from="1" position.to="32")
    constant(value=" Msg: ")
    property(name="msg" spifno1stsp="on")
    property(name="msg")
    constant(value="\n")
    }
```

Using the above template on a message would override the source severity to 2, or critical as defined by the Syslog protocol standard (RFC5424).

Rules

In Rsyslog, you can use rules to determine the flow and action of messages. A typical use case would be to send all messages Log Analytics, write some or all messages to local file, and then only send a subset of actionable messages as SNMP traps to Spectrum. Here is an example of a ruleset:

```
ruleset(name="network"){
# Send everything to LA and write to disk
      action(type="omfwd" protocol="tcp" target="LOGANALYTICSHOST" port="6514" template="ls_json")
if ($syslogpriority>=6) then {
    action(type="omfile" file="/var/log/networklog_debug" template="RSYSLOG_SyslogProtocol23Format")
        }
     else {
     action(type="omfile" file="/var/log/networklog" template="RSYSLOG_SyslogProtocol23Format")
# Drop messages we don't want to process at all
:msg, regex, "Command \"show.*" ~
:msg, regex, "%SNMPD-3-ERROR.*" ~
# Emergency syslogs
if (re_match($msg,"%[^:]*-0-[^:]*")==1) then {
        action(type="omsnmp" transport="udp" server="SPECTRUMSERVER" trapoid="1.3.6.1.4.1.19406.1.2.1"
port="162" version="1"
messageoid="1.3.6.1.4.1.19406.1.1.2.1" community="public" template="SNMPFormat")
    stop
        }
# Fatal syslogs
if (re_match($msg,"%[^:]*-1-[^:]*")==1) then {
        action(type="omsnmp" transport="udp" server="SPECTRUMSERVER" trapoid="1.3.6.1.4.1.19406.1.2.1"
port="162" version="1"
messageoid="1.3.6.1.4.1.19406.1.1.2.1" community="public" template="SNMPFormat")
    stop
        }
# NAT Address Alloc failures
      if (re_match($msg,".*%NAT-6-ADDR_ALLOC_FAILURE.*")==1) then {
        action(type="omsnmp" transport="udp" server="SPECTRUMSERVER" trapoid="1.3.6.1.4.1.19406.1.2.1"
port="162" version="1"
          messageoid="1.3.6.1.4.1.19406.1.1.2.1" community="public" template="SNMPFormat_Warning")
        stop
        }
}
```

In the simple example above, all messages are sent to Log Analytics, messages that are Informational or Debug are written to /var/log/networklog_debug, other messages are written to /var/log/networklog, messages with "Command show" and "%SNMPD-3-ERROR" are dropped from additional processing, and traps are being sent on any messages matching %*-0-*, %*-1-*, and %NAT-6-ADDR_ALLOC_FAILURE patterns with the NAT-6-ADDR_ALLOC_FAILURE messages being tagged as warning as opposed to the vendor defined informational.

Also to note, the version options for omsnmp are 0 and 1. These correspond to v1 and v2c, respectively. In the example above, SNMPv2c traps are being sent.

Example Configuration:

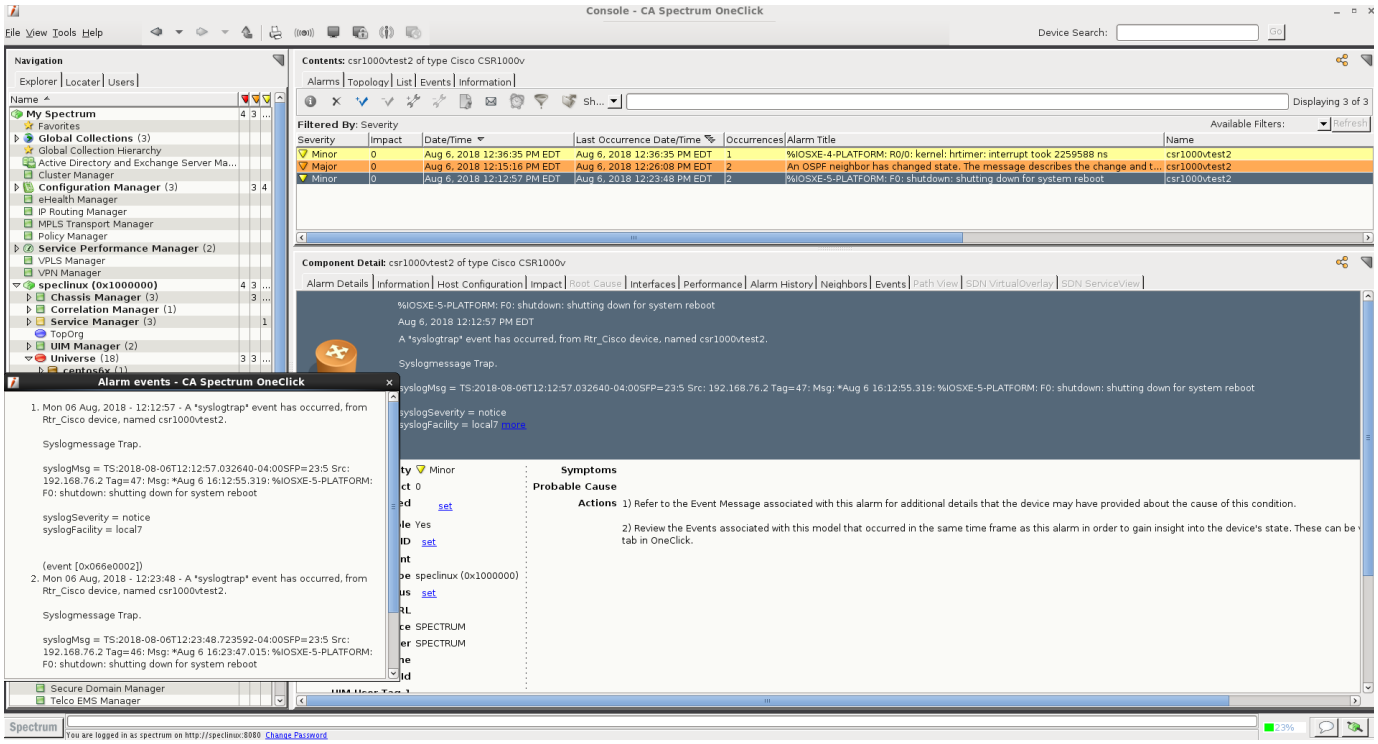
```
# rsyslog configuration file
```

```
# note that most of this config file uses old-style format,
# because it is well-known AND quite suitable for simple cases
# like we have with the default config. For more advanced
# things, RainerScript configuration is suggested.

# For more information see /usr/share/doc/rsyslog-*/rsyslog_conf.html
# If you experience problems, see http://www.rsyslog.com/doc/troubleshoot.html

#### MODULES ####


module(load="imuxsock") # provides support for local system logging (e.g. via logger command)
module(load="imklog")   # provides kernel logging support (previously done by rklogd)
#module(load"immark")  # provides --MARK-- message capability

module(load="omsnmp")

#template(name="forwardFormat" type="string" string="<%PRI%>%TIMESTAMP:::date-rfc3339% %fromhost-ip%
%syslogtag:1:3$)

#$ActionFileDefaultTemplate forwardFormat


# Provides UDP syslog reception
# for parameters see http://www.rsyslog.com/doc/imudp.html

module(load="imudp") # needs to be done just once
input(type="imudp" port="514")
#input(type="imudp" port="514" ruleset="network")


# Provides TCP syslog reception
# for parameters see http://www.rsyslog.com/doc/imtcp.html

#module(load="imtcp") # needs to be done just once
#input(type="imtcp" port="514")


#### GLOBAL DIRECTIVES ####


# Use default timestamp format
#$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
#$ActionFileDefaultTemplate RSYSLOG_ForwardFormat
$ActionFileDefaultTemplate RSYSLOG_SyslogProtocol23Format
#$ActionFileDefaultTemplate forwardFormat


# File syncing capability is disabled by default. This feature is usually not required,
# not useful and an extreme performance hit
#$ActionFileEnableSync on


# Include all config files in /etc/rsyslog.d/
$IncludeConfig /etc/rsyslog.d/*.conf

template(name="SNMPFormat" type="list") {
    constant(value=" TS:")
    property(name="timestamp" dateFormat="rfc3339")
    constant(value="SFP=")
    property(name="syslogfacility")
    constant(value=":")
    property(name="syslogpriority")
    constant(value=" Src: ")
    property(name="fromhost-ip")
    constant(value=" Tag=")
    property(name="syslogtag" position.from="1" position.to="32")
    constant(value=" Msg: ")
    property(name="msg" spifno1stsp="on")
    property(name="msg")
    constant(value="\n")
    }
```

```
template(name="SNMPFormat_Warning" type="list") {
    constant(value=" TS:")
    property(name="timestamp" dateFormat="rfc3339")
    constant(value="SFP=")
    property(name="syslogfacility")
    constant(value=":4 Src: ")
    property(name="fromhost-ip")
    constant(value=" Tag=")
    property(name="syslogtag" position.from="1" position.to="32")
    constant(value=" Msg: ")
    property(name="msg" spifno1stsp="on")
    property(name="msg")
    constant(value="\n")
    }

#### RULES ####


# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console


# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages


# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure


# Log all the mail messages in one place.
mail.*                                                  /var/log/maillog

# Log cron stuff
cron.*                                                  /var/log/cron


# Everybody gets emergency messages
*.emerg                                                 :omusrmsg:*


# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler


# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log

# ### begin forwarding rule ###
# The statement between the begin ... end define a SINGLE forwarding
# rule. They belong together, do NOT split them. If you create multiple
# forwarding rules, duplicate the whole block!
# Remote Logging (we use TCP for reliable delivery)
#
# An on-disk queue is created for this action. If the remote host is
# down, messages are spooled to disk and sent when it is up again.
#$WorkDirectory /var/lib/rsyslog # where to place spool files
#$ActionQueueFileName fwdRule1 # unique name prefix for spool files
#$ActionQueueMaxDiskSpace 1g   # 1gb space limit (use as much as possible)
#$ActionQueueSaveOnShutdown on # save messages to disk on shutdown
#$ActionQueueType LinkedList   # run asynchronously
#$ActionResumeRetryCount -1    # infinite retries if host is down
# remote host is: name/ip:port, e.g. 192.168.0.1:514, port optional
#*.* @@remote-host:514
# ### end of the forwarding rule ###
ruleset(name="network"){
if ($syslogpriority>=6) then {
    action(type="omfile" file="/var/log/networklog_debug" template="RSYSLOG_SyslogProtocol23Format")
```

```
        }
     else {
     action(type="omfile" file="/var/log/networklog" template="RSYSLOG_SyslogProtocol23Format

")
# Drop messages we don't want to process at all
:msg, regex, "Command \"show.*" ~
:msg, regex, "%SNMPD-3-ERROR.*" ~
# Emergency syslogs
if (re_match($msg,"%[^:]*-0-[^:]*")==1) then {
        action(type="omsnmp" transport="udp" server="SPECTRUMSERVER" trapoid="1.3.6.1.4.1.19406.1.2.1"
port="162" version="1"
messageoid="1.3.6.1.4.1.19406.1.1.2.1" community="public" template="SNMPFormat")
    stop
        }
# Fatal syslogs
if (re_match($msg,"%[^:]*-1-[^:]*")==1) then {
        action(type="omsnmp" transport="udp" server="SPECTRUMSERVER" trapoid="1.3.6.1.4.1.19406.1.2.1"
port="162" version="1"
messageoid="1.3.6.1.4.1.19406.1.1.2.1" community="public" template="SNMPFormat")
    stop
        }
# NAT Address Alloc failures
      if (re_match($msg,".*%NAT-6-ADDR_ALLOC_FAILURE.*")==1) then {
        action(type="omsnmp" transport="udp" server="SPECTRUMSERVER" trapoid="1.3.6.1.4.1.19406.1.2.1"
port="162" version="1"
          messageoid="1.3.6.1.4.1.19406.1.1.2.1" community="public" template="SNMPFormat_Warning")
        stop
        }
}
```

The integration included here maps the trap OID and varbind to a Spectrum event. That event then forwards to existing 0x3e00009 event which invokes Spectrum's ParseMap functionality to take advantage of 12K+ out of the box mappings. If a ParseMap entry does not exist, rather than following the out of the box message of indicating such, the 0x3dc0004 event is modified to call additional events which include procedures to extract variables from the message, including source, timestamp, facility, and severity and asserts an event/alarm of appropriate criticality on the source model:



Integration Package Contents (version 2 - download):

-rw-rw---- spectrum/spectrum 1181 2018-08-06 12:11 custom/Events/RSyslog/EventDisp

-rw-rw---- spectrum/spectrum 381 2018-04-06 22:45 custom/Events/RSyslog/AlertMap

-rw-r--r-- spectrum/spectrum 28 2017-02-17 08:06 custom/Events/RSyslog/SeverityMaps/SyslogSeverity

-rw-r--r-- spectrum/spectrum 431 2018-04-06 16:28 custom/Events/CsEvFormat/Event066e0000_en_US

-rw-r--r-- spectrum/spectrum 268 2016-09-16 08:18 custom/Events/CsEvFormat/Event066e0001_en_US

-rw-r--r-- spectrum/spectrum 228 2018-04-06 23:16 custom/Events/CsEvFormat/Event066e0002_en_US

-rw-r--r-- spectrum/spectrum 493 2016-04-07 14:39 custom/Events/CsPCause/Prob066e0002_en_US

Extract into $SPECROOT, update the event configuration, and once you have Syslog messages configured to come into Spectrum via the method mentioned above, you should be all set. Enjoy.