Pyrule для FM который пригодится каждому.

Есть такая ситуация, в нашей сети используются L3 свичи, маршрутизация на них устроена таким образом что в одном влане на каждом свиче настроен ip интерфейс и включен протокол маршрутизации, в данном случае EIGRP. Беда в том что когда пропадает связь с одним из свичей то в зависимости от размера сегмента, может возникнуть от 5 до 30 алармов, если например из-за проблем с питанием на узле, несколько свичей падают, то в FM возникает до сотни бессмысленных алармов. Решение простое, непонятно почему до сих пор не реализовано в апстриме.

```
from noc.fm.models import *
from noc.inv.models import *

@pyrule

def set_root_for_eigrp(alarm):

    eigrp = AlarmClass.objects.get(name="Network | EIGRP | Neighbor Down")
    alarms = ActiveAlarm.objects.filter(alarm_class=eigrp.id)
    ipif = SubInterface.objects.filter(managed_object=alarm.managed_object.id, enabled_afi=["IPv4"])

    for a in alarms:
        for i in ipif:
            if a.vars['neighbor']==i.ipv4_addresses[0].split("/")[0]:
                a.set_root(alarm)
```

Поясню логику на всякий случай. Этому пирулю задается интерфейс IAlarmTrigger, а в Alarm Trigger прописываем этот пируль для алармов Alarm Class RE: "NOC \| Managed Object \| Ping Failed". Что происходит:

1) Падает свич. через 15 секунд соседи обнаруживают не доступность и присылают сообщения в нок

2) В течение минуты нок тоже понимает, что свич недоступен, поднимает аларм "Ping failed"

3) По этому аларму срабатывает триггер и выполняется пируль

4) В пируль передается этот аларм ping failed. из него узнаем какая железка стала недоступной и для этой железки получаем список всех ее интерфейсов с ipv4 адресами, этот список кладется в ipif

5) Два цикла, перебираем все алармы EIGRP Neighbor Down и смотрим не относятся ли они к одному из интерфейсов нашей железки, если так то указываем наш изначальный аларм как причину (root cause) для каждого падения eigrp