Pyrule для Inventory, который пригодится для FM каждого

Заметил что народ в irc часто спрашивает как можно игнорировать алармы приходящие с портов куда включены клиенты. Действительно, кому нужны тысячи аварий каждодневно сваливающихся, потому что человек выключил свой компьютер. Но при всем при этом нас продолжают интересовать проблемы с сетевыми линками, ведь очевидно что обрыв на сети затронет не одного, а множество юзеров. И самое главное, ни в коем случае нельзя делать настройки руками, сделанное вручную имеет свойство устаревать и стоновится неактуальным очень быстро, нужна автоматика. Решение есть.

Чтобы реализовать мою задумку, нужно:

1) Включить любой метод discovery для обнаружения линков в NOC

2) Включить interface-discovery

3) Создать профили для сетевых и клиентских интерфейсов

4) Написать pyrule и настроить его выполнение

1 Включение discovery.

Тут все просто. В "Service Activation > setup > Managedobject profile" надо включить любой подходящий для конкретной сети метод дискавери. Очень желательно выбрать самый подходящий метод или несколько методов, потому что все будет зависить от того на сколько надежно будут находиться линки между устройствами.

2 Включение interface-discovery.

Все там же, в MO profile включаем interface-discovery. Так noc будет ходить по железкам и собирать все интерфейсы и укладывать их в базу. Если подумать, то это должно идти нулевым пунктом, потому что без интерфейсов, дискаверить линки не получится

3 Создание профилей

В "Inventory > setup > Interface Profiles" надо создать пару профилей:

первый назовем UNI - профиль который будет устанавливаться для клиентских портов. Для него в пункте Link Events устанавливаем значение "Ignore Link Events" или "Log events, do not raise alarms", в зависимости от того, хочется ли хранить историю падения клиентских портов (а иногда это важно для разбирательств)

второй назовем NNI - профиль для внутрисетевых интерфейсов. Для него в Link Events надо выбрать "Raise alarms", это значит что если придет event Link Down с указанием на этот порт то будет поднят аларм.

4 Pyrule

Вот тут начинается самое интересное. Создаем pyrule:

name: my_set_interface_profile

interface: IPeriodicTask

description: Set interface profile

text:

```
from noc.inv.models import *

@pyrule
def set_interface_profile(timeout = None):
    uni_id = InterfaceProfile.objects.get(name="UNI").id
    nni_id = InterfaceProfile.objects.get(name="NNI").id
    default_id = InterfaceProfile.objects.get(name="default").id
    interface = Interface.objects.filter(type="physical")
    for i in interface:
        if i.link:
            i.profile = nni_id
            i.save()
            continue
        elif not i.description:
            i.profile = default_id
            i.save()
            continue
        elif i.profile.name == InterfaceProfile.objects.get(name="default").name:
            i.profile = uni_id
            i.save()
            continue
    return True
```

Вот такой вот незамысловатый код. Логика тут простая, выдергиваем все физические порты что есть в базе и начинаем их перебирать. Если у интерфейса есть линк на другую железку то устанавливаем ему профиль NNI и записываем это значение в базу, если у интерфейса нет дескрипшена, значит считаем что порт не настроен и ставим ему профиль default, и самый конец, все что не попало под предыдущие два условия объявляются клиентскими, но с небольшим условием, если у порта стоит профиль default. Проверка профиля в последнем случае нужна вот для чего, если у вас есть какие-то свои профили портов, которые вы повесили на интерфейсы, этот пируль их не будет перезаписывать. Если клиентский порт вдруг окажется сетевым, то ему поднимется статус до сетевого без проблем и алармы будут приходить, если же сетевой порт станет клиентским, то придется вручную поправить, но опыт говорит что такое происходит категорически редко (лично я такого еще не делал).

Теперь остается добавить выполнение этого pyrule по расписанию. "Main > setup > schedules" Add schedule, где в Periodic task выбираем этот pyrule и устанавливаем время как часто его дергать, лично мне хватает раз в два часа (7200 секунд).

Вот и все, теперь noc будет самостоятельно следить за актуальностью профилей на портах и поднимать алармы только по важным сетевым событиям