ПоЛезные правила.

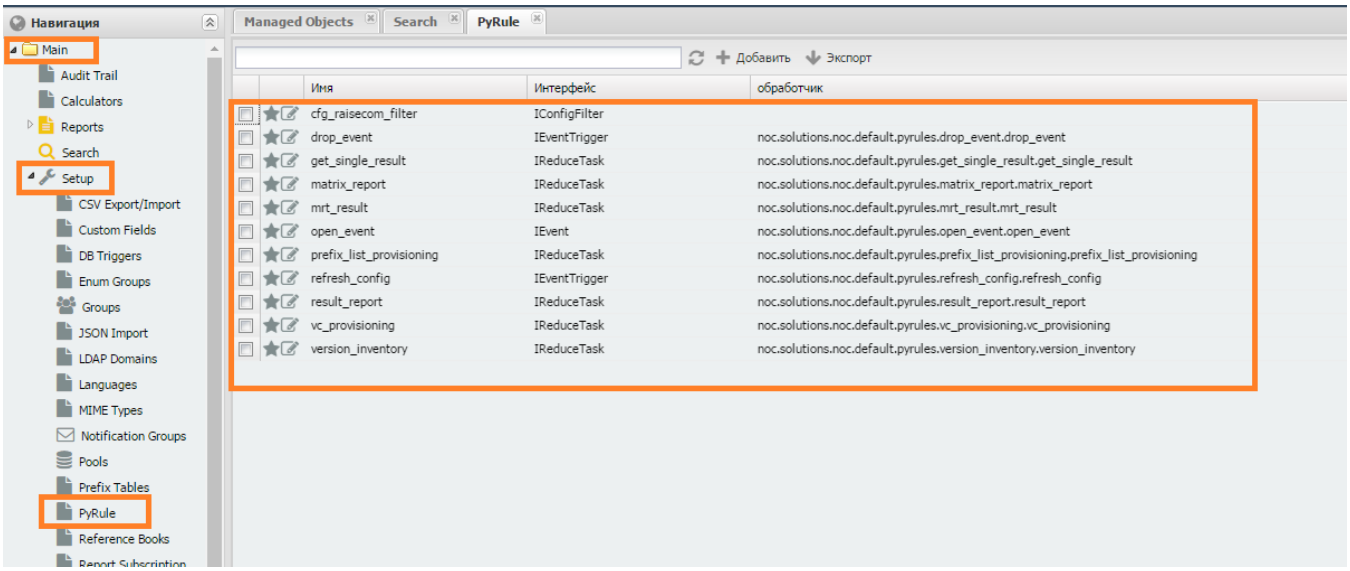
PyRule

Время от времени, пользователь НОКа страдает питоном. Время от времени, страдания приводят его к некоторому коду, который позволяет ему проводить больше времени вне ковыряния НОКа. За чтением какой-нибудь книжки, например. Вот этот самый код на питоне, который может делать что-то с НОКом и был названи PyRule или ПируЛем.

До микросервисной ветки это был хороший способ сделать что-то, чего НОК ещё делать не умел или не очень хотел. В микросервисной ветке функционал был чуть порезан но всё равно остаётся весьма широким.

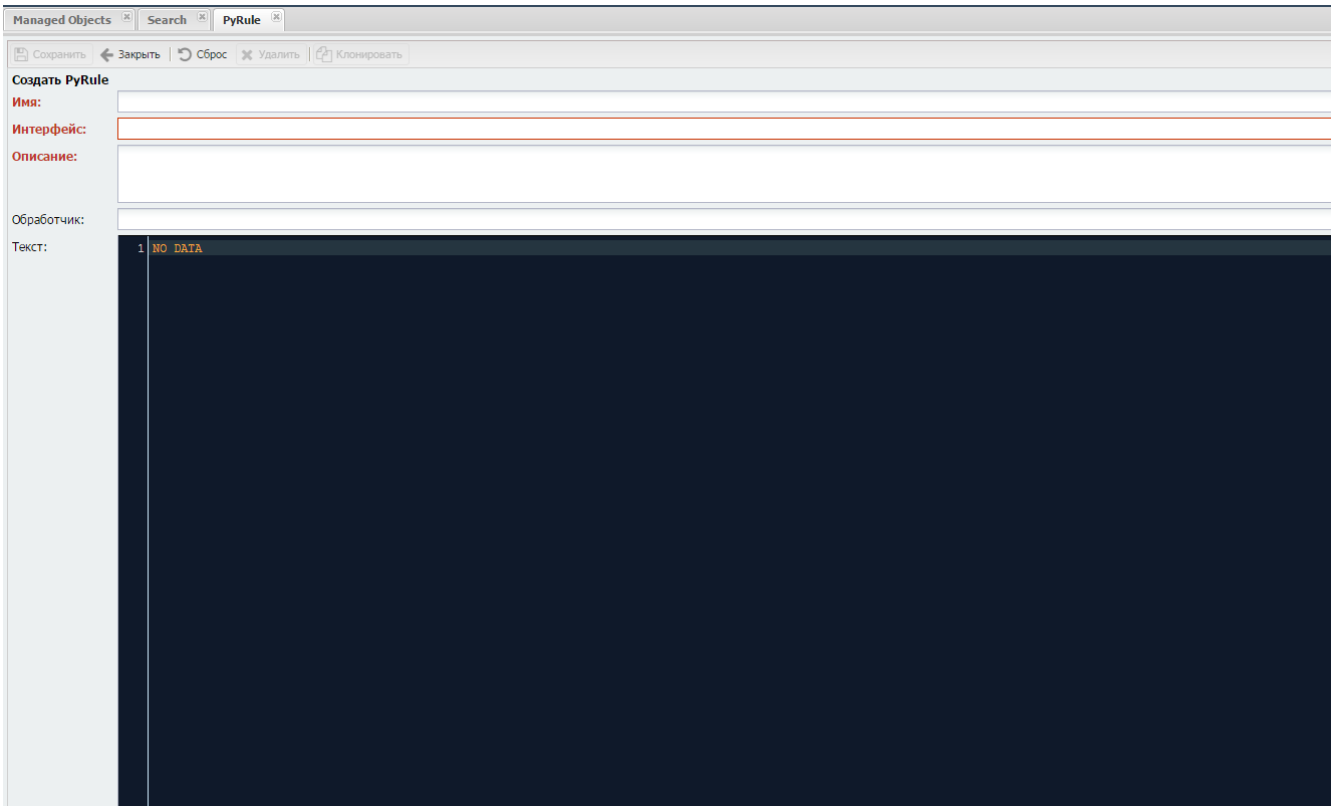
Создаём

Эти самые пирули обитают в меню Main->Setup->PyRule основного интерфейса.



Изначально, присутствуют только системные, но, можно добавить и свои. Своё делается по кнопке Create (Добавить).

Откроется меню по добавлению PyRule. Осветим основные поля доступные для заполнения.



Name (Имя)

Название в латинице. Будет показываться в интерфейсе.

Интерфейс (Interface)

Не тот, который есть в сетевых железках, но, принципы работы схожи. В этом меню выбирается интерфейс по которому пируль будет взамодействовать с НОКом. Этот самый интерфейс описывает, что мы получаем на вход, и что должны отдать на выход. Сами интерфейсы лежат в папочке **sa/interfaces**/. Свои можно ложить в папочку **custom/sa/interfaces/**, чтобы их не потёрло при деплое.

Разберём несколько интерфейсов, для примера:

IConfigFilter

```
class IConfigFilter(BaseInterface):
    managed_object = InstanceOfParameter("ManagedObject")
    config = StringParameter()
    returns = StringParameter()
```

В окошке выше написано следующие: На вход нам прилетает ManagedObject в виде объекта и конфи, в виде текста. На выход (returns) мы должны выдать некий текст.

IEvent

```
class IEvent(BaseInterface):
    event = InstanceOfParameter("Event")
```

В окошке выше сказано, что к нам прилетает Event (Событие). А на выход от нас ничего не ждут....

Description (Описание)

Описывает назначение PyRule

Handler (Обработчик)

Некоторый внешний файл на python, который запустится, когда произойдёт вызов PyRule

Текст

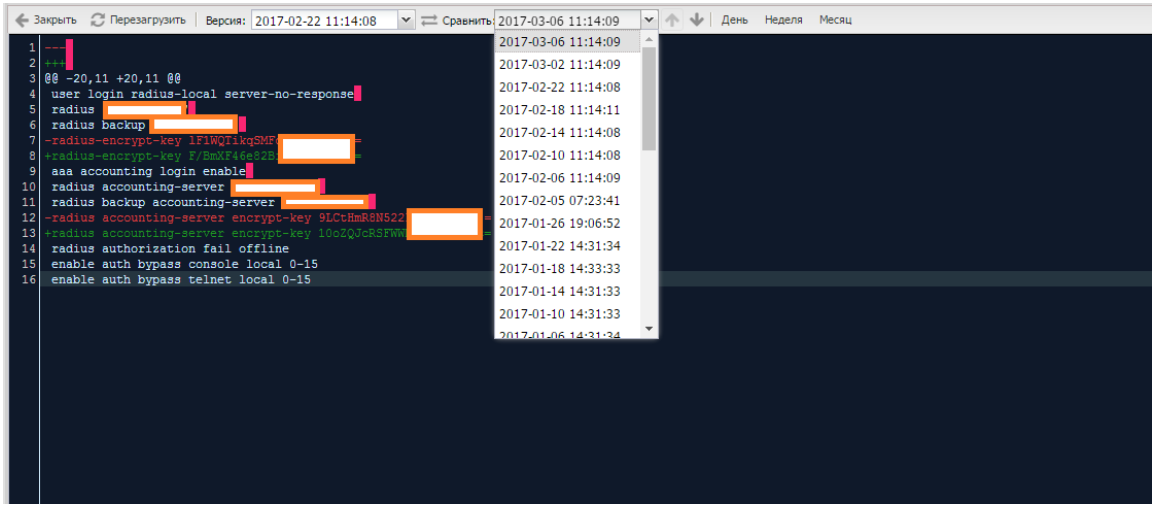
Собственно, текст самого пируля. Если задан handler (обработчик), можно ничего не писать.

Полезности

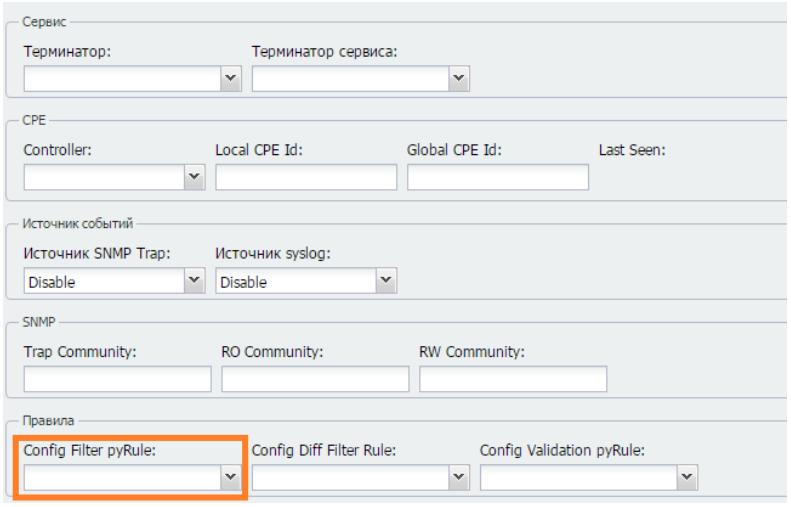
Собственно, для чего PyRule может быть полезен.

Фильтрация конфигурации

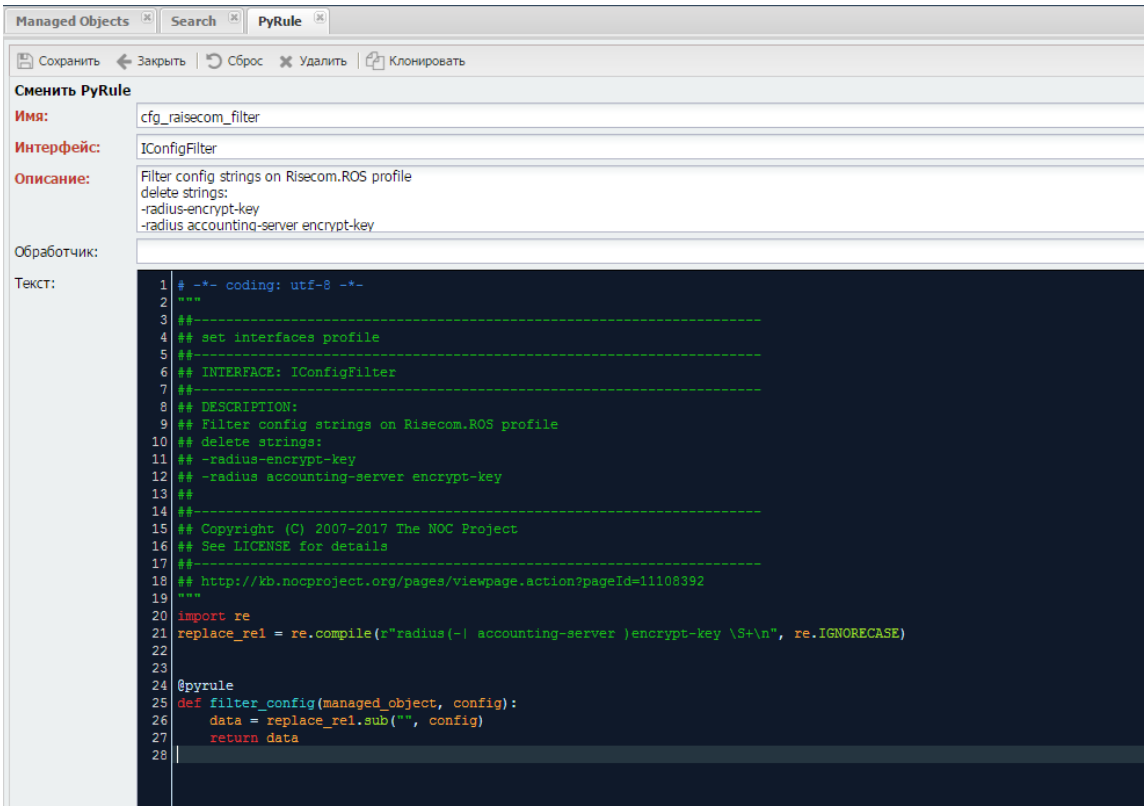
Как известно, НОК сохраняет только разницу в конфигурации между считываниями. Это удобно во многих случаях. Но, встречаются вредные железки, которые хранят в конфигурации н-р, ключи шифрования, или проставляют там дат изменения каждый день. В общем, всячески пишут туда информацию, которую там видеть бы не хотелось. Н-р



Хотелось бы, убрать эти строчки из конфигурации. Чтобы не мешали жить. Как раз по этому поводу есть несколько полей в конфигурации ManagedObject:



В выделенном поле можно выбрать пируль, который запустится перед сравнением конфигураций и, соответственно, у нас есть шанс отрезать лишнее.
В качестве лишних мы определим строчки с ключами, на скриншот выше. Пируль будет следующего вида:



ConfigFilter PyRule

```
# -*- coding: utf-8 -*-
"""
##----------
## set interfaces profile
##----------
## INTERFACE: IConfigFilter
##----------
## DESCRIPTION:
## Filter config strings on Risecom.ROS profile
## delete strings:
## -radius-encrypt-key
## -radius accounting-server encrypt-key
##
##----------
## Copyright (C) 2007-2017 The NOC Project
## See LICENSE for details
##----------
## http://kb.nocproject.org/pages/viewpage.action?pageId=11108392
"""
import re
replace_re1 = re.compile(r"radius(-| accounting-server )encrypt-key \S+\n", re.IGNORECASE)

@pyrule
def filter_config(managed_object, config):
    data = replace_re1.sub("", config)
    return data
```

Он просто вырезает лишние строчки из конфигурации.

Полезности от пользователей

Ссылка	Интерфейс	Примечание
Pyrule для FM который пригодится каждому.	IAlarmTrigger	
Pyrule для Inventory, который пригодится для FM каждого	IPeriodicTask	В микросервисах убрали Шедулер из веб-интерфейса

Это, как бы, нельзя.