


Используем API для автоматизации работы с Proxmox Mail Gateway

 interface31.ru/tech_it/2021/11/ispol-zuem-api-dlya-avtomatizacii-raboty-s.html

Записки IT специалиста

Технический блог специалистов ООО "Интерфейс"

- [Главная](#)
- Используем API для автоматизации работы с Proxmox Mail Gateway

Proxmox Mail Gateway - бесплатный, но гибкий и мощный пограничный почтовый шлюз от компании Proxmox, более известной своими средствами виртуализации. Как и остальные продукты, он выделяется простым и удобным веб-интерфейсом, позволяющим быстро и удобно настраивать продукт. Но как быть, если нужно выполнить большое количество однотипных действий? Либо интегрировать шлюз с каким-либо иным используемым вами ПО? В таком случае вам следует использовать API - специальный программный интерфейс, открывающий широкие возможности для автоматизации.



Онлайн-курс по устройству компьютерных сетей

На углубленном курсе "[Архитектура современных компьютерных сетей](#)" вы с нуля научитесь работать с Wireshark и «под микроскопом» изучите работу сетевых протоколов. На протяжении курса надо будет выполнить более пятидесяти лабораторных работ в Wireshark.

ССС

Как и многие другие наши статьи, эта появилась благодаря обратной связи. Один из наших коллег спросил: а можно ли как-то автоматизировать добавление объектов в белый список? Список относительно небольшой - 150-200 email-адресов и десятков-другой доменов. Но забивать его руками - то еще удовольствие, кроме того, списки следует время от времени дополнять и этот процесс тоже хотелось бы автоматизировать.

Здесь мы вплотную подошли к понятию **API** (*application programming interface*) - программного интерфейса приложения, это набор высокоуровневых способов взаимодействия с приложением без необходимости вникать во внутреннее устройство продукта. API, как правило, хорошо документированы и остаются постоянными, либо сохраняют обратную совместимость, на всем жизненном цикле ПО. Это позволяет создать инструмент автоматизации единственный раз и не

беспокоиться о том, что разработчики что-то серьезно поменяют в его внутреннем устройстве, разве что только расширять возможности, если в API появятся новые функции.

Proxmox Mail Gateway имеет достаточно обширный и развитый API, который полностью описан в документации. На первый взгляд она может показаться довольно сложной, но на самом деле разобраться в ней несложно, даже если вы делаете это в первый раз.

Слева представлено древовидное представление всех доступных через API объектов, а справа показаны доступные с ними действия. Работать с API можно как по протоколу HTTP, так и через интерфейс командной строки, в данной статье мы будем рассматривать именно последний способ.

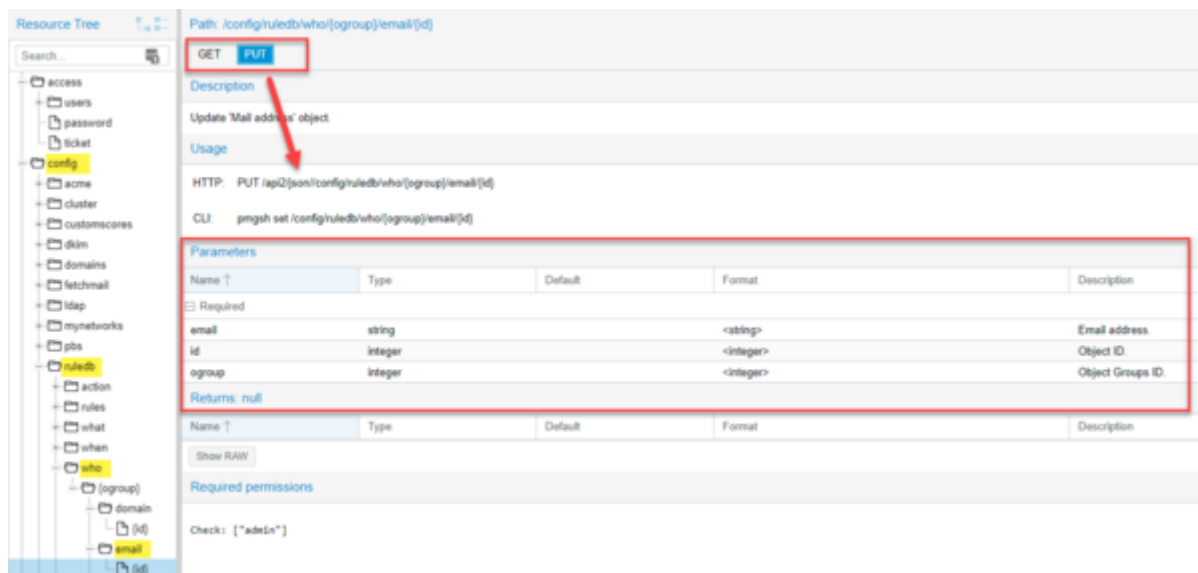
Вернемся к нашей задаче, раскроем последовательно пункты **config - ruledb - who** и найдем объект **email**, справа мы увидим его состав: строковое поле **email** для хранения почтового адреса и целочисленный **ogroup** - идентификатор группы объекта. Также нам доступно действие **create**, т.е. создание объекта.

The screenshot shows the Proxmox Mail Gateway API interface. On the left is a 'Resource Tree' with a search bar and a list of resources. The 'config' resource is expanded, showing 'ruledb' and 'who'. Under 'who', the 'email' resource is selected. The main panel shows the configuration for the 'email' object. The path is '/config/ruledb/who/{ogroup}/email'. The description is 'Add Mail address object'. The usage section shows HTTP and CLI commands. The parameters section is highlighted with a red box and labeled 'Поля объекта' (Object fields). It contains a table with columns: Name, Type, Default, Format, and Description. The table has two rows: 'email' (string) and 'ogroup' (integer). The 'Returns' section shows 'integer'. The 'Required permissions' section shows 'Check: ["addIn"]'.

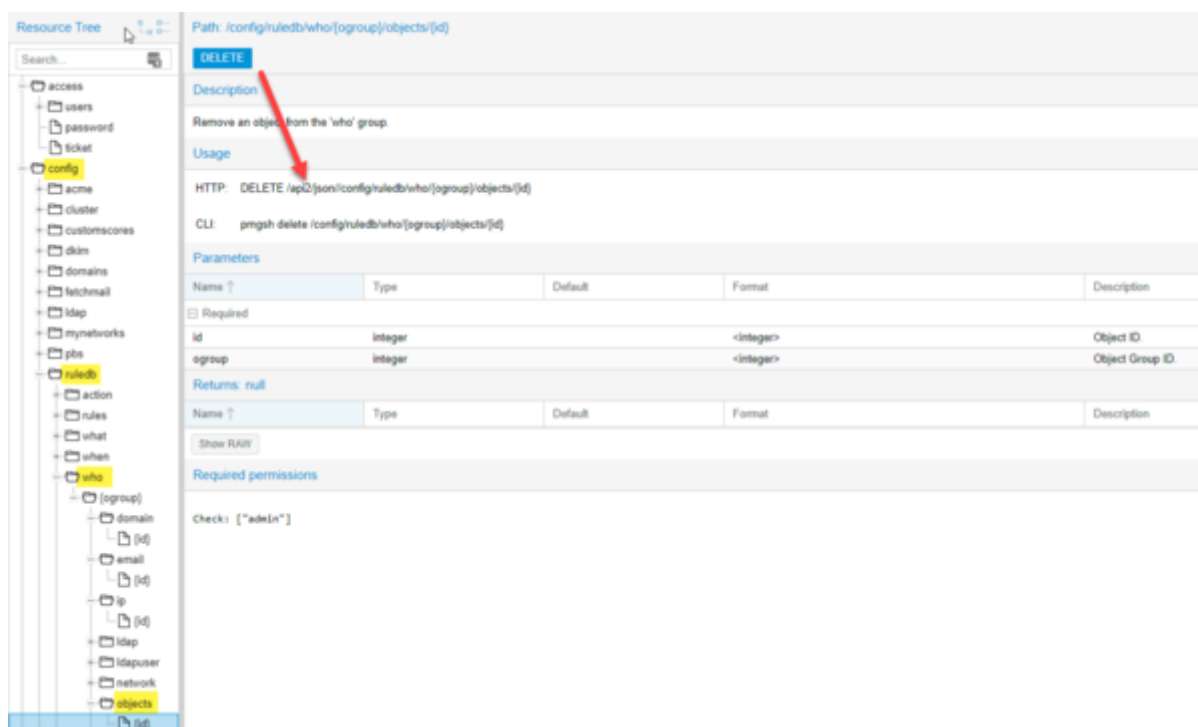
Name	Type	Default	Format	Description
email	string		<string>	Email address
ogroup	integer		<integer>	Object Groups ID

Если мы перейдем к уже созданному объекту, то увидим, что в его составе появилось еще одно целочисленное поле - **id** - идентификатор объекта. Чтобы не путаться в дальнейшем давайте уделим идентификаторам несколько минут. Разворачивая в дереве папку **who** мы увидим вложенную в нее папку **{ogroup}**, которая уже содержит все нужные нам объекты. Что такое **{ogroup}**? Это объект-владелец, если мы заглянем в веб-интерфейс, то увидим, что внутри объекта **Who** уже созданы два списка, черный и белый, каждый из них имеет свой **id**, который будет являться **ogroup** для всех нижестоящих объектов, т.е. показывать их принадлежность к тому или иному списку.

Для объекта электронного адреса нам доступно два действия: **get** - получить данные объекта и **set** - установить новые значения. Например, мы можем изменить электронный адрес записи, если вы вдруг ошиблись при вводе.



А вот если мы захотим удалить какую-либо запись, то нам придется пойти другим путем. Для этого не нужно уточнять тип записи, достаточно знать ее идентификатор и идентификатор группы владельца.



Как видим - все довольно несложно, если есть желание, то разобраться в структуре и методах работы с API можно довольно быстро. Тем более что примеры использования приведены на каждой странице, нас сейчас интересуют те, которые помечены как CLI, по сути, это готовые команды, все что вам остается сделать, это правильно подставить в них идентификаторы.

Для работы через интерфейс командной строки предназначена специальная утилита **pmgsh**, есть два варианта ее использования: можно использовать ее как команду, каждый раз передавая ей параметры, либо запустить как отдельную консоль и работать с ней интерактивно, последний вариант, если вы работаете руками, более удобен, так как в нем будет действовать автодополнение по **Tab**.

Чтобы войти в отдельную консоль достаточно запустить **pmgsh** без параметров, при этом изменится приглашение командной строки:

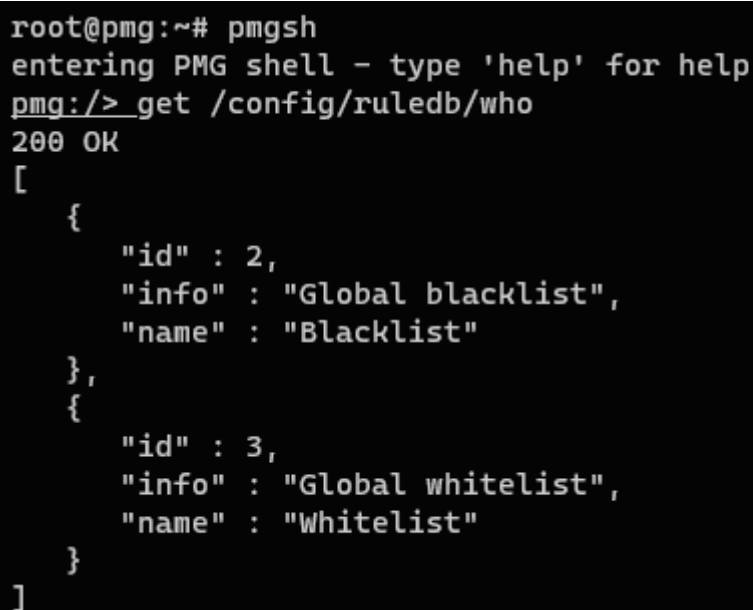
```
pmgsh
```

Теперь можно выполнять команды, давайте посмотрим какие объекты есть внутри **Who**:

```
get /config/ruledb/who
```

В качестве альтернативы можно выполнить, если вы по какой-то причине не хотите переходить в отдельную консоль, следующую команду:

```
pmgsh get /config/ruledb/who
```



```
root@pmg:~# pmgsh
entering PMG shell - type 'help' for help
pmg:./> get /config/ruledb/who
200 OK
[
  {
    "id" : 2,
    "info" : "Global blacklist",
    "name" : "Blacklist"
  },
  {
    "id" : 3,
    "info" : "Global whitelist",
    "name" : "Whitelist"
  }
]
```

В выводе мы увидим две предустановленные группы: глобальные белый и черный списки, а также их идентификаторы. Отдельно следует обратить внимание на формат вывода - это JSON, это специальный формат обмена данными, который легко читается людьми и широко поддерживается в различном ПО и языках программирования. Проще говоря, если вам нужно передать результат для автоматической обработки дальше, то это не составит особого труда. Ну а для того, кто знает, что такое JSON и как с ним работать - ничего вообще пояснять не нужно.

Узнав из вывода прошлой команды идентификатор белого списка получим его состав:

```
get /config/ruledb/who/3/objects
```

```
pmg:/> get /config/ruledb/who/3/objects
200 OK
[
  {
    "descr" : "mail@fromthisdomain.com",
    "email" : "mail@fromthisdomain.com",
    "id" : 2,
    "ogroup" : 3,
    "otype" : 1001,
    "otype_text" : "Mail address",
    "receivertest" : 0
  }
]
```

Пока в нем находится один единственный адрес, добавленный разработчиками для примера. Еще одна тонкость, для получения состава списка мы использовали **objects**, потому как для отдельных видов записей, таких как **email**, **domain** или **ip** действие **get** недоступно, только **create**. Чтобы не путаться в доступных действиях всегда смотрите документацию, она должна стать вашей настольной книгой при работе с API Proxmox Mail Gateway.

Давайте попробуем что-нибудь создать, например, новый почтовый адрес:

```
create /config/ruledb/who/3/email --email ivanov@example.com
```

В данной команде мы указали **ogroup - 3**, т.е. белый список, затем уточнили, что создаем объект **email** и через двойной дефис указали добавляемый параметр и его значение. Затем можем снова запросить список объектов белого списка и увидеть, что в него добавлен еще один адрес.

```

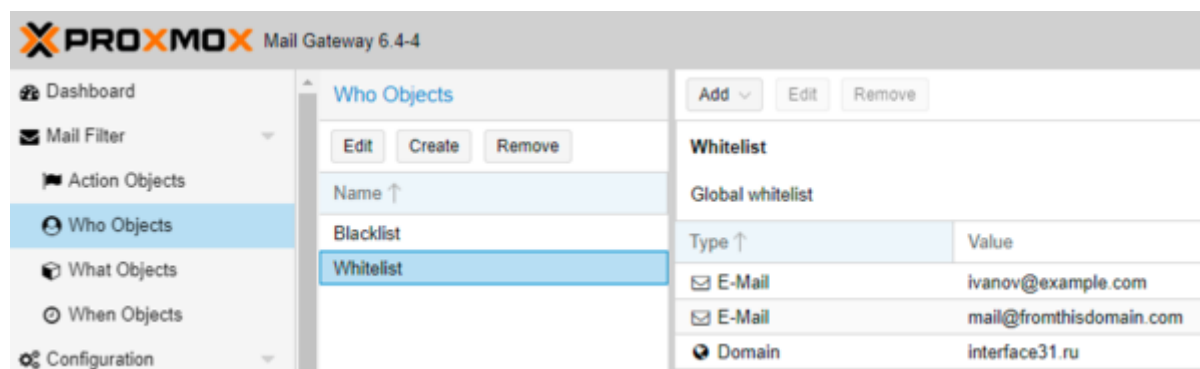
pmg:/>_create /config/ruledb/who/3/email --email ivanov@example.com
200 OK
46
pmg:/>_get /config/ruledb/who/3/objects
200 OK
[
  {
    "descr" : "ivanov@example.com",
    "email" : "ivanov@example.com",
    "id" : 46,
    "ogroup" : 3,
    "otype" : 1001,
    "otype_text" : "Mail address",
    "receivertest" : 0
  },
  {
    "descr" : "mail@fromthisdomain.com",
    "email" : "mail@fromthisdomain.com",
    "id" : 2,
    "ogroup" : 3,
    "otype" : 1001,
    "otype_text" : "Mail address",
    "receivertest" : 0
  }
]

```

Аналогичным образом мы можем добавить объект другого типа, скажем, домен:

```
create /config/ruledb/who/3/domain --domain interface31.ru
```

А теперь заглянем в веб-интерфейс, где увидим в составе белого списка все добавленные нами через API элементы.



Создавать объекты мы уже научились, теперь попробуем удалить. Допустим мы хотим удалить адрес добавленный для примера, для этого нам нужно знать его идентификатор и идентификатор группы, все это есть на скриншотах выше: **id - 2**, **ogroup - 3**. Располагая данной информацией выполним команду:

```
delete /config/ruledb/who/3/objects/2
```

Если теперь проверить состав белого списка, то мы увидим там только внесенные нами записи.

```
pmg:/>_delete /config/ruledb/who/3/objects/2
200 OK
pmg:/>_get /config/ruledb/who/3/objects
200 OK
[
  {
    "descr" : "ivanov@example.com",
    "email" : "ivanov@example.com",
    "id" : 46,
    "ogroup" : 3,
    "otype" : 1001,
    "otype_text" : "Mail address",
    "receivertest" : 0
  },
  {
    "descr" : "interface31.ru",
    "domain" : "interface31.ru",
    "id" : 47,
    "ogroup" : 3,
    "otype" : 1002,
    "otype_text" : "Domain",
    "receivertest" : 0
  }
]
```

Теперь, когда вы уже немного освоились с API, перейдем к тому, ради чего мы все это затеяли - автоматизации. Хотя, возможно, найдутся и любители консоли, которым будет удобнее выполнять некоторые действия через API, нежели веб-интерфейс. Но у нас стоит простая, но утомительная задача - добавить в белый список некоторое количество записей, немного, всего сотню - другую.

Давайте для примера создадим аналог такого списка. Перейдем в домашнюю директорию и создадим новый файл, в который внесем необходимые нам электронные адреса:

```
cd ~
nano email.list
```

В данном примере мы использовали установленный по умолчанию в системе редактор **nano**, если у вас установлен **mc** и вам более по душе его редактор, то выполните:

```
mcedit email.list
```

```
GNU nano 3.2 email.list Modified
petrov@example.com
kozlov@example.org
smirnova@example.com
vasya@pupkin.mail

^G Get Help  ^O Write Out  ^V Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos   ^U Undo      ^M Mark Text
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell   ^_ Go To Line ^E Redo      ^G Copy Text
```

А теперь внесем этот список в Proxmox Mail Gateway и даже сможем обойтись без скриптов, выполнив в одну строку:

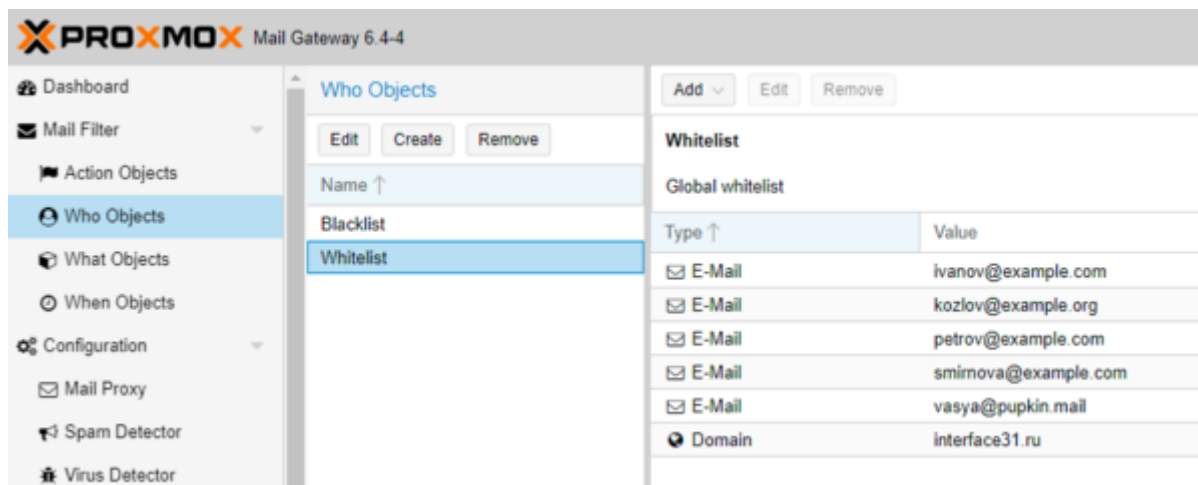
```
while read email ; do pmgsh create /config/ruledb/who/3/email --email "$email" ;
done < email.list
```

Результатом работы данной конструкции будет вывод идентификатора для каждой добавленной записи.

```
root@pmg:~# while read email ; do pmgsh create /config/ruledb/who/3/email --email "$email" ; done < email.list
200 OK
48
200 OK
49
200 OK
50
200 OK
51
```

То, что мы сделали, может показаться кому-то особой "консольной магией", но на самом деле все просто. Команда **while - do - done** запускает цикл, который будет работать до тех пор, пока на вход поступают строковые данные, внутри цикла мы обращаемся к API и записываем новый объект **email** для каждой прочитанной строки. После чего мы воспользовались перенаправлением потоков и передали на вход команде содержимое текстового файла.

Результат, как говорится, налицо. Никакого ручного труда, особенно если данный список мы тоже можем сформировать автоматически. Всю рутину делает за нас компьютер, оставляя время на действительно важные и серьезные задачи.



Данная статья не претендует на подробное руководство по работе с API, ее цель - познакомить вас с новым, мощным и удобным инструментом, а также показать все его преимущества на примере решения реальной задачи.

- **Категории:**

- Безопасность в сети,
- Сети и интернет,
- Системному администратору,
- Электронная почта

- **Теги:**

- API,
- E-mail,
- Proxmox,
- Автоматизация,
- Безопасность,
- Сетевые технологии