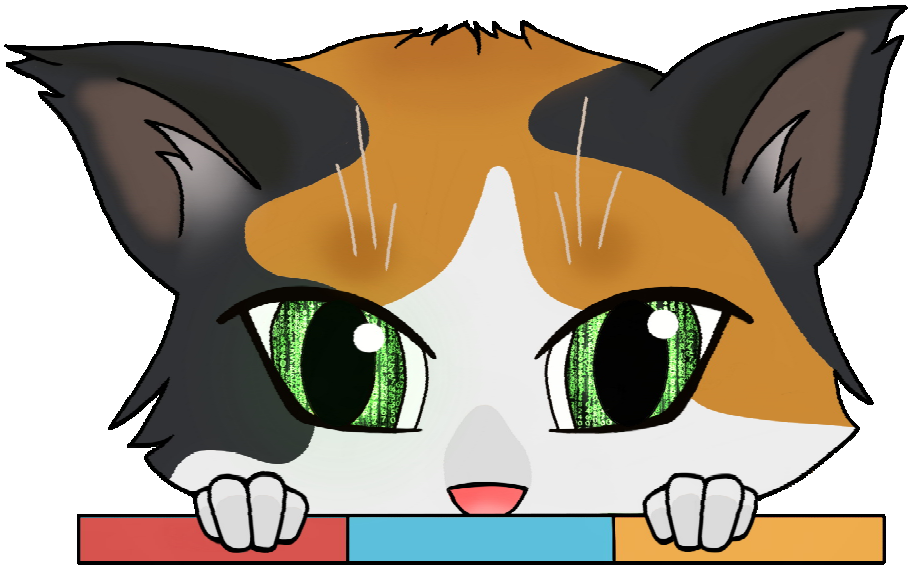
Network Systems Group

Богатка



Программно-определяемая глобальная сеть для массового подключения удалённых объектов

Руководство администратора

Версия 1.3.11

Обновлено 12.06.2024

Москва 2024

§ СОДЕРЖАНИЕ §

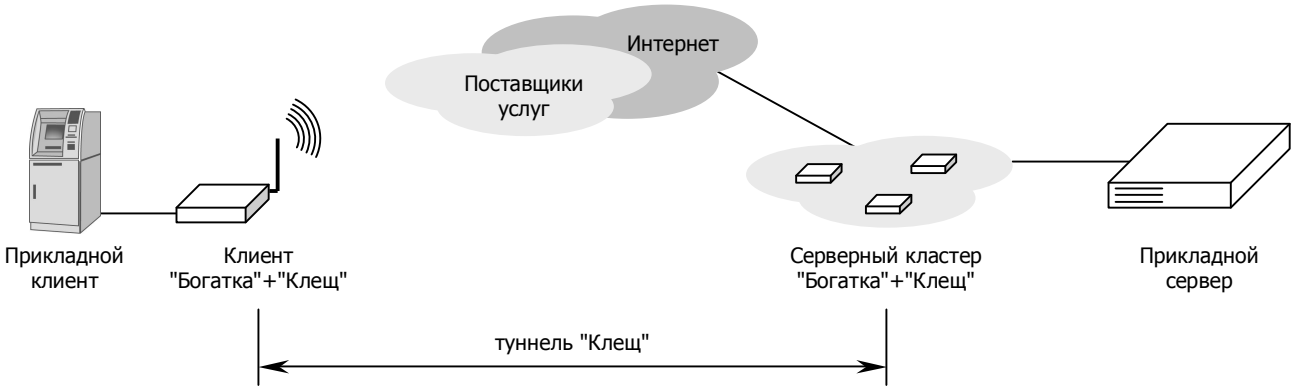
ООО «ЭН–ЭС–ДЖИ»
Россия 105187 Москва
ул. Вольная, д.35
Тел./факс: (495) 727–19–59 (многоканальный)

http://www.nsg.ru/
mailto:info@nsg.net.ru
mailto:sales@nsg.net.ru
mailto:support@nsg.net.ru

§1. Общие сведения и топология системы

§1.1. Общие понятия и определения

Основной задачей системы является передача трафика IP (Layer 3) или Ethernet (Layer 2) между *прикладным сервером*, расположенным в корпоративной сети заказчика, и *прикладными клиентами*, расположенными за *клиентскими устройствами* (CPE) системы. При этом серверная часть системы размещается также внутри корпоративной сети, либо на её границе, а доступ клиентов к ней осуществляется по сетям общего пользования. Для интеллектуального управления потоками данных, а также для защиты данных, используется фирменная технология туннелирования NSG "Клещ".



Вторичными задачами являются:

— Автоматическая настройка туннелей при добавлении новых клиентов.
— Мониторинг работы и управление, в т.ч. массовое, клиентскими устройствами.

Специфика этих задач состоит, в данном случае, в наличии большого числа клиентских устройств с однотипными аппаратными комплектациями и программными конфигурациями.

ВНИМАНИЕ В рамках данной системы используется следующая терминология:
— Клиент — *устройство*, установленное на удалённой площадке.
— Туннель — соединение между клиентским устройством и серверным кластером.
— Пользователь — *человек*, имеющий доступ к системе.
— Администратор — пользователь, имеющий полные права доступа в системе.

Вся работа по установке и администрированию системы "Богатка" производится в серверной части, поэтому данный документ относится по существу к ней. Настройка и администрирование клиентских устройств осуществляются в рабочем порядке в ходе эксплуатации системы.

Серверная часть системы реализована в виде *виртуальных контейнеров* Docker. Узлы (*nodes*) Docker, на которых они исполняются, как правило, являются *физическими хостами*, но в общем случае могут быть и виртуальными машинами. Предполагается, что читатель данного документа знаком с администрированием сетей, серверов Linux и контейнеров Docker. По этой причине общие вопросы перечисленных технологий в документе не рассматриваются. Кратко резюмируются лишь ключевые моменты, существенные для установки и эксплуатации системы.

Функциональной единицей системы являются контейнеры с серверной частью приложения "Богатка" и туннелей "Клещ". Далее они будут называться *серверами "Богатка"*. Реализация приложения, описываемая в данной редакции документа, выполняет двоякую функцию и решает одновременно основную и обе вспомогательные задачи, указанные выше.

В штатной конфигурации системы на каждом узле работает один сервер "Богатка". Одновременно с ними на узлах системы могут исполняться другие контейнеры, входящие в состав системы (подробнее см. ниже):

— База данных с информацией о туннелях, их конфигурациях и инвентаризационных данных (при небольшом числе узлов — рекомендуется на каждом узле).
— Сервер Zabbix (на одном ведущем узле).
— База данных Zabbix с накопленной телеметрией о работе системы (на этом же узле).
— Web-интерфейс Zabbix (на этом же узле).
— Рекомендуется: агент Zabbix для мониторинга самого узла (на каждом узле).

Информация о клиентах, их конфигурациях, инвентаризационные данные и другая статическая информация хранится в *базе данных* "Богатка" (в отдельном контейнере PostgreSQL). На каждом узле работает, как правило, своя копия БД, синхронизация между ними производится автоматически. При этом одна из копий (*главная*) в каждый момент времени открыта на чтение и запись, все остальные — только на чтение. В случае отказа этой БД автоматически назначается новая главная копия. Объём хранимой информации невелик, её перезапись происходит относительно редко (только при изменении состава клиентов или их конфигурации), поэтому для ускорения работы с БД рекомендуется использовать на хостах в качестве основного носителя SSD — как для ОС самого хоста, так и для контейнеров.

Информация о текущем состоянии клиентов и принятая от них телеметрия хранится только в оперативной памяти контейнеров "Богатка", синхронизируется между ними, но не записывается на постоянные носители. Для хранения и анализа она пересылается на сервер *Zabbix*, который хранит её в своей долговременной БД, выводит в виде графиков, диаграмм и отчётов за выбранные периоды времени, и т.п. Сервер Zabbix функционирует в отдельном контейнере на одном из узлов. "Богатка" использует сервер Zabbix следующим образом:

— Создаёт на нём клиентов, группы и другие объекты мониторинга в соответствии с тем, что имеется в ней самой, изменяет и удаляет их при помощи WebAPI Zabbix.
— Данные, интересующие заказчика, собираются на клиенте и отправляются на сервер Богатка в формате сообщений Zabbix (*traps*). Для этого используется служба обработчика событий и действие #zabbixsender (в командном дереве NSG, см. services.event-handler) и/или утилита zabbix-sender (в скриптах *bash*). Сервер служит в качестве ретранслятора и пересылает эти данные на сервер Zabbix от своего имени.
— Запрашивает заданную информацию при помощи WebAPI Zabbix, обрабатывает её и выводит результаты в своём пользовательском интерфейсе.

ПРИМЕЧАНИЕ Разработкой ООО "ЭН–ЭС–ДЖИ" является только контейнер nsgru/bogatka . Остальные контейнеры, используемые в составе системы, являются продуктами с открытым кодом от сторонних разработчиков. ООО "ЭН–ЭС–ДЖИ" не несёт ответственности за их содержание, функциональные качества, устойчивость работы, безопасность и др.

Физические хосты в системе целесообразно разделить на *обычные* и *ведущие*. Ведущие хосты отличаются от обычных следующими дополнительными функциями:

— Сервера Zabbix. Для хранения БД Zabbix — объёмной и постоянно перезаписываемой в ходе работы — хост должен быть оснащён HDD или дисковым массивом, в дополнение к системному SSD. В текущей реализации предусмотрено использование только одного сервера Zabbix.
— Сервера Syslog. Поскольку уж ведущий хост оснащён HDD, то целесообразно оправлять на него сообщения Syslog со всех обычных хостов, чтобы не расходовать ёмкость и ресурс их собственных SSD. Серверов Syslog в системе может быть несколько одновременно.
— Копия БД клиентов на этом узле, как правило, используется в качестве главной по умолчанию.

Отказ ведущего хоста не приводит к утрате основной функциональности системы и грозит, в самом худшем случае, только потерей данных Zabbix и Syslog. Безусловно, ведущие хосты целесообразно располагать ближе к центру системы и рабочему месту администратора, и именно их желательно собирать на базе высоконадёжных серверных комплектующих, с резервируемым питанием, и т.п. Обычные хосты допустимо, в целях экономии, собирать из комплектующих потребительского класса, поскольку разовые отказы серверов предусмотрены архитектурой системы и резервирование обеспечивается на уровне кластера в целом.

§1.2. Сетевые порты

Для работы системы используется протокол TCP. Номера портов TCP устанавливаются переменными окружения (параметрами -e) в команде docker run ... , которая выполняется вручную или, что удобнее, вызывается из скриптов, заранее подготовленных для конкретной инсталялции (см. §2.2–§2.5).

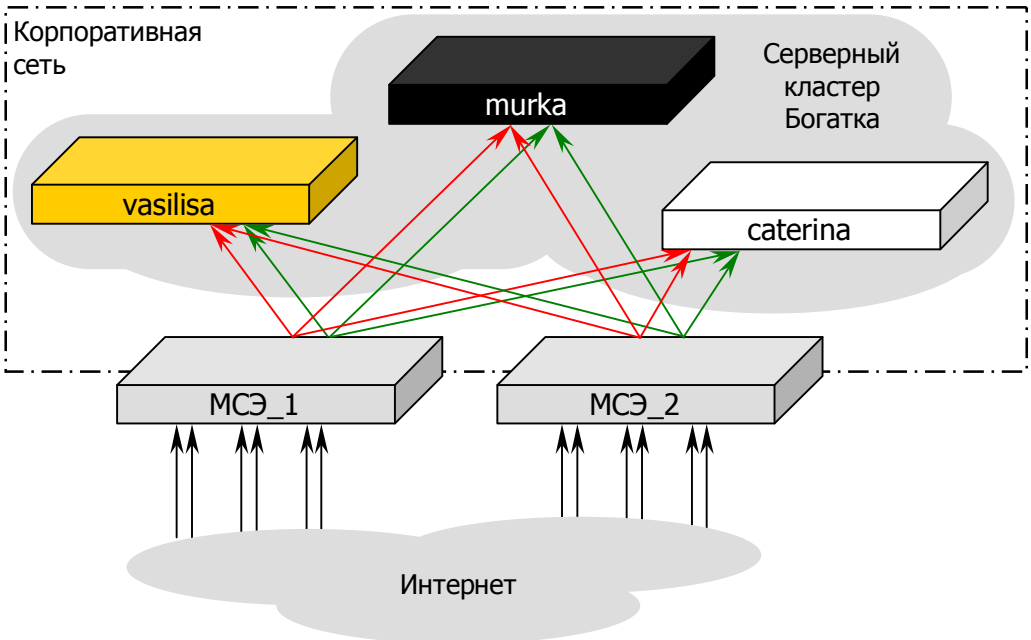
Два порта необходимы для работы клиентов по существу:

— Порт для доступа к Web API "Богатка". По этому порту клиенты запрашивают конфигурацию у сервера. По существу, для получения конфигурации используется HTTPS со специфическим форматом запроса. Порты для конфигурации клиентов и для доступа пользоватей в Web-интерфейс разделены по соображениям безопасности.
Номер данного порта (или внешнего, NAT-ируемого на него) необходим для заводской конфигурации клиентов "Богатка", специфичной для данного заказчика.

— Порт для туннелей (с шифрованием TLS или без него, по усмотрению заказчика). Этот порт указывается в рабочей конфигурации клиентов которую они получают от сервера.

Если система "Богатка" находится внутри корпоративной сети, скрытой за входным маршрутизатором (межсетевым экраном), то на нём необходимо настроить пробос, как минимум, двух портов TCP (Destination NAT) с внешнего интерфейса на внутренний IP-адрес каждого из серверов "Богатка". Внешние номера портов при этом могут быть любыми по усмотрению заказчика.

ПРИМЕЧАНИЕ Количество серверов в кластере "Богатка" никак не связано с количеством точек входа в корпоративную сеть. Если тех и других несколько (обычно именно так и должно быть, по соображениям надёжности), то, в общем случае, в каждой точке входа должен быть настроен DNAT своей пары портов на каждый из серверов "Богатка".



При искусственном разделении и ограничении доступа (например, корпоративная сеть состоит из сегментов в разных городах, в каждом сегменте есть свои точки входа и свои сервера "Богатка"), то DNAT настраивается в каждой точке входа на каждый из серверов, который должен быть доступен через неё.

Следующие два порта обязательно должны быть доступны в пределах корпоративной сети. Пробрасывать доступ на эти порты из сетей общего пользования следует только в случае, если предусматривается работа пользователей извне корпоративной сети:

— Порт для доступа к Web-интерфейсу системы по HTTPS (на каждом узле).
— Порт для доступа к Web-интерфейсу Zabbix (на ведущем узле).

Остальные порты используются только в пределах кластера "Богатка" внутри корпоративной сети:

— Порт для доступа к Web API Zabbix (на ведущем узле).
— Порт для приёма сообщений Zabbix (на ведущем узле).
— Порт для обращения к БД "Богатка" (на каждом узле, на котором работает копия БД).

Помимо перечисленных, требуется также доступ со стороны серверов Богатка к серверам NSTP, ГИС и некоторых других служб, которые могут располагаться в сетях общего пользования или внутри корпоративной сети. Подробнее см. §2.6. Если эти службы находятся в сети общего пользования, а сервера "Богатка" — в корпоративной сети, то для доступа достаточно имеющегося механизма Source NAT или Masquerading, поскольку все обращения к этим службам инициируются со стороны Богатки.

§1.3. Резервирование физических хостов

Архитектура систем "Богатка" и "Клещ" предусматривает резервирование ядра сети по принципу N+1. Это означает, что весь кластер серверов обслуживает весь массив клиентов. Балансировка нагрузки между серверами производится автоматически. Если для заданного количества клиентов требуется, исходя из производительности, N физических хостов, то для резервирования следует добавить в систему один хост. В штатном режиме все сервера работают с недогрузкой на 1/(N+1)-ю долю. Если один из хостов выходит из строя, то остальные перераспределяют между собой его клиентов и работают с полной нагрузкой.

В особо ответственных системах практикуется резервирование по схеме N+2, N+3, или исходя из допустимого количества хостов, которые гипотетически могут выйти из строя одновременно. (Например, количество резервных хостов равно максимальному числу хостов, размещённых на одной площадке, на случай, если эта площадка будет выведена из строя целиком в результате техногенной катастрофы или стихийного бедствия.)

Нецелесообразно использовать резервирование по схеме 1+1, т.е. разбивать массив клиентов на части, каждая из которых обслуживается отдельной парой серверов (1 основной + 1 резервный). Такая топология не только требует удвоенного числа серверов (фактически это не 1+1, а N×(1+1)), умноженных вдвое расходов, умноженных в N раз трудозатрат, но и потенциально более уязвима при возможном выходе из строя более чем 1 сервера.

§1.4. Права и область ответственности пользователей

Предполагается, что в структурированной иерархической модели управления обязанности и ответственность распределены следующим образом:

— *Администраторы* отвечают за те части системы, которые непосредственно влияют на работу каждого её элемента: сервера, шаблоны, классы, а также группы и пользователей.

— Администраторы также отвечают за критически ответственные операции, такие как массовое обновление ПО или конфигурации.

— *Пользователи с полными правами* отвечают за работу клиентов, которые входят в состав назначенных им групп.

— Пользователи не могут выполнять действия, которые могут повлиять на работу других клиентов, пользователей или системы в целом.

— *Пользователи с ограниченными правами* выполняют вспомогательную роль. Это могут быть, например, стажёры, которым необходимо время для знакомства с системой. Это могут быть также полноправные пользователи, которым, помимо их основных групп, поручено попутно, "одним глазом", присматривать за смежными группами на случай возникновения каких-либо ситуаций, требующих немедленного реагирования.

Например, создание новых клиентов доступно только администраторам, поскольку им ещё не назначена никакая группа. Но после того, как новый клиент создан и включён в одну из групп, всю дальнейшую работу с ним — индивидуальную настройку, эксплуатацию, устранение неисправностей — целесообразно делегировать владельцу группы, освободив от этого администраторов.

§1.5. Web API

Web API позволяет сторонним приложениям получать данные из системы "Богатка" посредством запросов HTTP/HTTPS. При наличии соответствующих методов, возможна также передача данных из внешних приложений в систему. Для работы Web API используются те же порты TCP, что и для ручного доступа в интерфейс системы.

Полное описание текущей версии представлено в документе NSG:

SD–WAN "Богатка". Web API. Руководство разработчика.

§2. Установка и перезапуск системы

§2.1. Установка ПО узла

Узлом системы Docker может являться как физический хост, так и виртуальная машина. Операции, специфические для физических хостов, оговорены особо.Часть операций 1–5 может быть уже выполнена заранее, например, в рамках услуги аренды виртуальных серверов.

1. Установить ОС и последние обновления. Рекомендуется последняя версия Ubuntu Server LTS, но администратор системы вправе выбирать другой дистрибутив Linux по своему усмотрению. Клавиатуру и локаль можно оставить только латинские, в кириллице необходимости нет. Сразу установить openssh-server. При установке достаточно создать единственного обязательного пользователя, другие не требуются. Обратить внимание на то, чтобы имена хостов различались (инсталлятор может дать им одинаковые имена по умолчанию.)

Настроить сеть (если она не была настроена сразу как надо при установке системы). Пример: в Ubuntu Server сетевые настройки хранятся в файлах /etc/netplan/*.yaml (имя файла может варьироваться, но в свежеустановленной системе это, как правило, единственный файл).

```
$ cat /etc/netplan/50-cloud-init.yaml
network:
  ethernets:
    eno1:
      addresses:
      - 10.33.22.11/8
      gateway4:
        10.0.23.45
      nameservers:
        addresses:
        - 10.0.56.78
    eno2:
      dhcp4: true
      optional: true
version: 2
```

Для неподключённых и неиспользуемых портов существенно optional: true, чтобы не ждать их готовности при старте системы.

2. Настроить виртуальный *swap*. Рекомендуется на машинах с SSD, чтобы не расходовать его ресурс записи-перезаписи.

```
$ sudo apt-get install zram-config
$ sudo systemctl start zram-config.service
```

Отключить своп-файл (устанавливается по умолчанию):

```
$ sudo swapoff /swap.img
```

и удалить или закомментировать строку swap ... в файле /etc/fstab . После этого файл /swap.img можно удалить. Рекомендуется рестартовать сервер и убедиться, что своп работает должным образом. Команда

```
$ swapon --show
```

должна показывать несколько блочных устройств, по числу ядер в системе, в совокупности занимающих половину физической памяти. (Это настройки по умолчанию, подробнее см. *man swapon*). Иных своп-устройств быть не должно.

3. Настроить часовой пояс и клиента NTP. В зависимости от используемого дистрибутива, может потребоваться отключить сервис systemd-timesyncd, используемый по умолчанию, и установить более точного клиента — ntpd. Пример для Ubuntu Server:

```
$ sudo dpkg-reconfigure tzdata
$ sudo timedatectl set-ntp no
$ sudo apt-get install ntp ntpdate
```

Проверить результат с помощью команд systemctl status и timedatectl. (Формат вывода может зависеть от версии. Подробнее см. справку по данным командам.)

Если система будет работать в закрытой корпоративной сети и использовать внутренний сервер NTP вместо общедоступных, настроенных по умолчанию, то отредактировать файл /etc/ntp.conf и перезапустить сервис:

```
$sudo systemctl daemon-reload
$sudo systemctl restart ntp.service
```

Через 5 минут с помощью команды ntpq -p проверить явным образом, с какими серверами синхонизируется данный хост.

4. На ведущем сервере (-ах) разметить HDD (одним разделом) и смонтировать его в выбранную директорию (ниже в данном руководстве — прямо в /mnt). Создать для него запись в fstab . (Если система и логи пишутся в один раздел, то данный пункт не требуется.)

5. На ведущем хосте(-ах) включить и настроить сервер Syslog. На него будут писаться журналы всех хостов и контейнеров. Пример для Ubuntu Server:

По умолчанию установлен rsyslog. В файле /etc/rsyslog.conf раскомментировать строки в разделе Modules:

```
module(load="imudp")
input(type="imudp" port="514")
```

Настроить правила логирования в разные журналы в /etc/rsyslog.d/*.conf .

Рестартовать rsyslog:

```
$ sudo systemctl restart rsyslog.service
```

Проверить с помощью netstat -na | grep 514 и logger, что сервер работает.

На обычных хостах настроить отправку syslog на ведущий хост (или несколько).

6. Сгенерировать и разместить ключи и сертификаты:

На каждом узле создать директорию /etc/ssl/bogatka . Поместить в неё закрытый ключ (ssl.key) и сертификат (ssl.crt) сервера. Они используются для работы HTTPS и TLS (загрузка конфигурации, Web-доступ для пользователей, WebAPI, туннели "Клещ") как в контейнерах Богатка, так и в контейнерах Zabbix, дабы не сотворять излишних сущностей. Пример для быстрой генерации ключей и самоподписанного сертификата:

```
$ sudo openssl req -new  -newkey rsa:2048 -sha256 -x509 -days 3650 -nodes -out ssl.crt \
  -keyout ssl.key -config /etc/ssl/openssl.cnf \
  -subj /C=RU/ST=Russia/L=Moskvabad/O=MyPoopyBank/OU=IT/CN=bogatka/
```

Сгенерировать сертификат также можно любой из утилит для организации удостоверяющего центра: XCA, TinyCA, GnoMint и т.п. Для рабочих инсталляций рекомендуется использовать реальный сертификат, выданный организации-заказчику вышестоящим удостоверяющим центром.

Для файла ssl.key установить права на чтение для всех пользователей, чтобы к нему могли обращаться пользователи из контейнеров Богатка и Zabbix:

```
$ sudo chmod 644 /etc/ssl/bogatka/ssl.key
```

Помимо них, на ведущих серверах для работы Zabbix необходим файл параметров Диффи-Хеллмана, сгенерированный в соответствии с принятой политикой безопасности, например:

```
$ sudo openssl dhparam -outform PEM -out dhparam.pem -5 2048
```

ВНИМАНИЕ В контейнерах Zabbix имена файлов ключа и сертификата не настраиваются, поэтому должны точно соответствовать указанным.

7. Установить Docker Engine — Community в соответствии с инструкцией:

https://docs.docker.com/install/linux/docker-ce/ubuntu/

Дать пользователю узла права на работу с Docker (включить его в группу docker):

```
$ sudo usermod -aG docker имя_пользователя
```

Изменение вступает в силу после следующего входа данного пользователя в систему.

8. Установить инструментарий для работы с Git и дополнительные утилиты:

```
$ sudo apt install git
```

9. Рекомендуется также сразу установить всякие полезные мелочи

```
$ sudo apt install net-tools traceroute whois mc ...
```

Рекомендуется настроить проверку файловой системы при каждом старте системы. Это несколько увеличивает время старта сервера, но в идеале он должен работать непрерывно в течение всего срока эксплуатации, поэтому наиболее частая причина рестарта — это временное отключение электропитания, и в этом случае файловая система может быть повреждена.

```
$ sudo tune2fs -c 1 /dev/sda2
```

где sda2 — корневой раздел системы (откорректировать в зависимости от инсталляции).

10. Создать рабочую директорию и клонировать в неё публичный репозиторий NSG:

```
$ mkdir ~/opt
$ cd ~/opt
$ git clone https://github.com/nsg-ru/bogatka-docker
$ cd bogatka-docker/host-mode
```

Дальнейшая установка и запуск системы производится с помощью скриптов, находящихся в данной директории. Скрипты загружают нужные контейнеры с ресурса *https://hub.docker.com* и запускают их. Для лучшего понимания рекомендуется ознакомиться с содержимым указанных скриптов и с *man docker*. Параметры скриптов перечислены ниже.

Для всех новых инсталляций рекомендуется устанавливать контейнеры "Богатка" в режиме хоста, как определено скриптами в данной директории. В этом режиме они будут непосредственно иметь доступ к сетевым интерфейсам узла. Это упрощает топологию сети, облегчает её понимание и настройку. Установка с использованием роя, рекомендованная ранее для предыдущих версий Богатки, более не является целесообразной.

§2.2. Установка кластера баз данных

1. Определить имена БД в данной инсталляции "Богатка". Рекомендуется использовать имена в формате, стандартном для контейнера bitnami/postgresql: pg-0, pg-1 и т.д.
2. Отредактировать скрипты для работы с БД: db-run, db-migrate, db-seed, db-show . В скриптах используются следующие переменные, которые должны быть указаны в соответствии с фактическими именами:

   NODES — Список всех БД, в произвольном порядке. Список одинаков для всех узлов. Рекомендуется сразу включить в него все планируемые БД, даже если они в данный момент ещё не созданы, чтобы в будущем не приходилось корректировать скрипты и рестартовать БД. Несуществующие БД будут просто рассматриваться как временно недоступные, это штатная ситуация.

   NAME — Имя БД на данном узле. Уникальное для каждого узла.

PRIMARY Имя главной БД. Эта БД должна создаваться первой и указываться одинаково для всех последующих. Если она доступна, то именно она будет открыта на чтение и запись, все остальные — только на чтение. Если она становится недоступной, то остальные БД выбирают новую главную.
При добавлении новых контейнеров БД в процессе эксплуатации необходимо убедиться, что данный параметр указывает на текущую главную базу, поскольку она могла измениться с момента первоначальной установки системы.

3. Запустить контейнер с БД "Богатка":

```
$ sudo mkdir -p /var/lib/postgresql/docker/$NAME
$ sudo chown 1001:1001 /var/lib/postgresql/docker/$NAME
$ ./db-run
```

Здесь $NAME — как и в скриптах, имя БД на данном узле. 1001 — идентификатор пользователя, используемый в контейнере БД (он фиксирован и не настраивается). Существуют ли пользователь и группа с таким ID на хосте и как они называются — не имеет значения; но если они существуют, то такие пользователи получают полный доступ к этой директории.

ПРИМЕЧАНИЕ Скрипт db-run не может использоваться как готовый для *пере*запуска БД в процессе эксплуатации, поскольку не известно, какая из копий окажется ведущей в этот момент — она может отличаться от указанной в PRIMARY. Для перезапуска БД необходимо сначала определить текущую главную БД и отредактировать скрипт (см. §2.8).

4. Проверить, что контейнер запустился:

```
$ docker ps
CONTAINER ID  IMAGE                      COMMAND                  CREATED             STA...
0258f89dc10d  bitnami/postgresql:latest  "/entrypoint.sh /run..."  About a minute ago  UP ...
```

5. Создать базу данных и все таблицы в ней (только один раз при установке первой БД):

```
$ ./db-migrate
```

6. Создать пользователя БД admin с паролем admin (только один раз при установке первой БД):

```
$ ./db-seed
```

7. Добавить разрешение всех имён, назначенных контейнерам БД, на внутренний DNS корпоративной сети либо в файлы /etc/hosts на всех узлах системы.

Работающий контейнер БД открывает на узле порт TCP, по умолчанию — 5432.

§2.3. Установка сервера Zabbix

Для установки сервера Zabbix следует выполнить скрипт:

```
$ ./zabbix-run
```

Скрипт содержит 3 команды docker run ... :

Первая — создаёт контейнер с БД Zabbix. Это отдельная база, она имеет другое содержимое, нежели БД "Богатка", другой объём и режим работы, поэтому она хранится отдельно. Из параметров команды следует обратить внимание и, вероятно, скорректировать по месту следующий:

```
-v /var/lib/postgresql/docker/zabbix-bogatka/data:/var/lib/postgresql/data
```

Здесь подчёркнутое — путь к директории БД Zabbix на физическом хосте. Если, как рекомендуется, система хоста установлена на SSD, а HDD используется исключительно для хранения больших баз и журналов, то сюда следует подставить его точку монтирования и дальнейший путь в его файловой системе, например /mnt/zabbix-data .

Вторая — запускает контейнер Zabbix. В параметрах команды указана созданная выше БД, имя и пароль для неё.

ВНИМАНИЕ Web-интерфейс Zabbix принимает соединения по HTTPS на порту TCP 443. Этот порт не должен использоваться какими-либо другими службами на данном хосте.

По умолчанию, на сервере создаётся пользователь с правами суперадминистратора со стандартным именем Admin и стандартным паролем zabbix — что, естественно, недопустимо при реальной эксплуатации. После установки необходимо зайти на сервер по https://ад.рес.хос.та и изменить, в целях безопасности, как минимум, пароль. Рекомендуется установить максимально длинный и сложный пароль, чтобы его не пришлось изменять в будущем; в случае изменения пароля придётся редактировать скрипт для запуска контейнера "Богатка" и перезапускать его на каждом хосте.

Если корпоративная политика безопасности не допускает работу пользователей с типовыми именами, то следует создать для работы Богатки отдельного пользователя с уникальным именем. Для полной синхронизации данных между Богаткой и Zabbix необходимо, чтобы этот пользователь имел в Zabbix права суперадминистратора. В противном случае Богатка не сможет автоматически создавать в Zabbix свои группы узлов, и это превратится в ручной труд администратора.

Для подготовки сервера к работе необходимо выполнить следующие действия:

— Создать группу узлов сети с именем Bogatka_all
— Создать группу пользователей с именем Bogatka_all
— Импортировать шаблон Bogatka_client (файл содержится в архиве zabbix-setup.zip).
— Оформить глобальный экран Богатки и экраны клиентов (по усмотрению заказчика, примеры см. в том же архиве)

§2.4. Установка кластера "Богатка"

Работающий контейнер "Богатка" имеет непосредственный доступ к IP-интерфейсам узла и открывает на них нужные порты TCP. Помимо физических интерфейсов хоста, контейнер "Богатка" создаёт туннельные интерфейсы с именами вида tunNNN. Номера этих интерфейсов генерируются динамически при подключении клиентов, а их настройки определяются шаблоном для серверной стороны туннелей (см. п. §3.4).

Перед запуском контейнера необходимо отредактировать скрипт bogatka-host-run. Скрипт содержит, по существу, одну команду docker run ... , которая содержит большое число параметров и переменных окружения (-e ...), передаваемых контейнеру. Для удобства настройки скрипта, ключевые переменные, специфичные для каждой инсталляции, вынесены в отдельные переменные окружения *bash* в заголовке скрипта. Рекомендуется редактировать их, а не саму команду.

Переменная в контейнере	Переменная в скрипте	Описание
—	NAME	Имя контейнера "Богатка" на данном узле. Уникальное для каждого узла и не совпадающее с именем БД на нём. Рекомендуется использовать имя, совпадающее с *hostname* узла, дабы излишних сущностей не плодить.
Переменные для работы с БД. Нестандартные номера портов (отличные от 5432) можно указывать после имени через двоеточие, например, pg-1:5555.		
DB_HOSTS_RW	DB_RW	Список БД, потенциально доступных на запись, в соответствии с их фактическими именами в системе (через запятую). Первой в списке должна стоять главная БД, указанная в скрипте db-run в качестве PRIMARY. Если эта БД становится недоступной, то Богатка пробует запись во все последующие, пока не найдёт новую главную. При этом через каждые 5 минут она проверяет, не восстановилась ли эта копия в качестве главной.

DB_HOSTS_RO	DB_RO	Список БД, доступных на чтение, в соответствии с их фактическими именами в системе. Первой в списке должна стоять локальная копия на данном хосте (кроме очень больших инсталляций, где держать копию БД на каждом хосте может быть нецелесообразно).
Переменные для инициализации БД. Используются только один раз в скриптах db-migrate и db-seed .		
DB_HOST		Имя или IP-адрес хоста, на котором работает БД.
DB_PORT		Номер порта для подключения к БД. Порт, на котором БД ждёт обращений, устанавливается при запуске БД, подробнее см. документацию по контейнеру bitnami/postgresql .
Порты Богатки		
BG_HTTP_PORT	—	Номер порта HTTP. По умолчанию 0, то есть отключён.
BG_HTTPS_PORT	HTTPS_PORT	Номер порта HTTPS для пользовательского интерфейса, по умолчанию 50443.
BG_API_PORT	API_PORT	Номер порта HTTPS для обращения к Web API Богатки, по умолчанию 50444.
BG_TLS_PORT	ACARI_PORT	Номер порта для подключения клиентов по TLS, по умолчанию 50019.
BG_TCP_PORT	TCP_PORT	Номер порта для подключения клиентов без TLS, по умолчанию 50020.
Переменные для подключения к Zabbix и ГИС. Номера портов, на которых сервер Zabbix ждёт обращения, устанавливаются при запуске сервера, подробнее см. документацию по контейнерам zabbix/zabbix-web-nginx-pgsql и zabbix-server-pgsql .		
ZBX_EXT_URL	ZEU	Zabbix External URL — внешний URL для обращения к Web-интерфейсу Zabbix. Если сервер находится внутри корпоративной сети, скрытой за входным маршрутизатором, то здесь следует указать его доменное имя (разрешаемое через DNS) или внешний IP-адрес, а также номер внешнего порта TCP, который для которого настроен Destination NAT на порт HTTPS Zabbix (по умолчанию, 8443) ведущего узла. Данный URL будет использоваться в качестве HTML-ссылки, передаваемой пользователю для перехода из Web-интерфейса системы "Богатка" в Web-интерфейс Zabbix. Если доступ к Zabbix предполагается только из внутренней сети, то здесь можно указать внутренний IP-адрес или доменное имя ведущего узла. Пример: ZEU="https://demo.nsg.net.ru:10443"
ZBX_API_URL	ZAU	Zabbix API URL — URL для доступа к API Zabbix. Должен обязательно содержать имя и пароль суперадминистратора, назначенные после установки сервера (см. §2.3). Используется в пределах внутренней сети. Пример: ZAU="http://Admin:zabbix@zabbix-web-nginx-pgsql:8080" Номер порта по умолчанию 8443.
ZBX_SND_HOST	ZSH	Доменное имя или IP-адрес сервера Zabbix, которому передаётся телеметрия (*traps*). Пример: ZSH="zabbix-server-pgsql" Для контейнера на ведущем узле можно указать имя localhost или адрес 127.0.0.1 .
ZBX_SND_PORT	ZSP	Номер порта TCP для отправки телеметрии на сервер Zabbix. По умолчанию, 10051. Устанавливается в

		настройках контейнера Zabbix.
ZBX_LISTEN_PORT	ZLP	Номер порта TCP для приёма телеметрии от клиентов. По умолчанию, 50051. Указывается в конфигурации клиента Zabbix на клиентском устройстве.

Все используемые URL и имена контейнеров должны разрешаться в IP-адреса на соответствующем DNS (внешнем, либо внутреннем в сети заказчика), либо в файле /etc/hosts на каждом узле.

При необходимости, отредактировать остальные параметры в команде docker run ... В частности, следует обратить внимание на строки вида

```
-v /путь/на/узле:/путь/в/контейнере
```

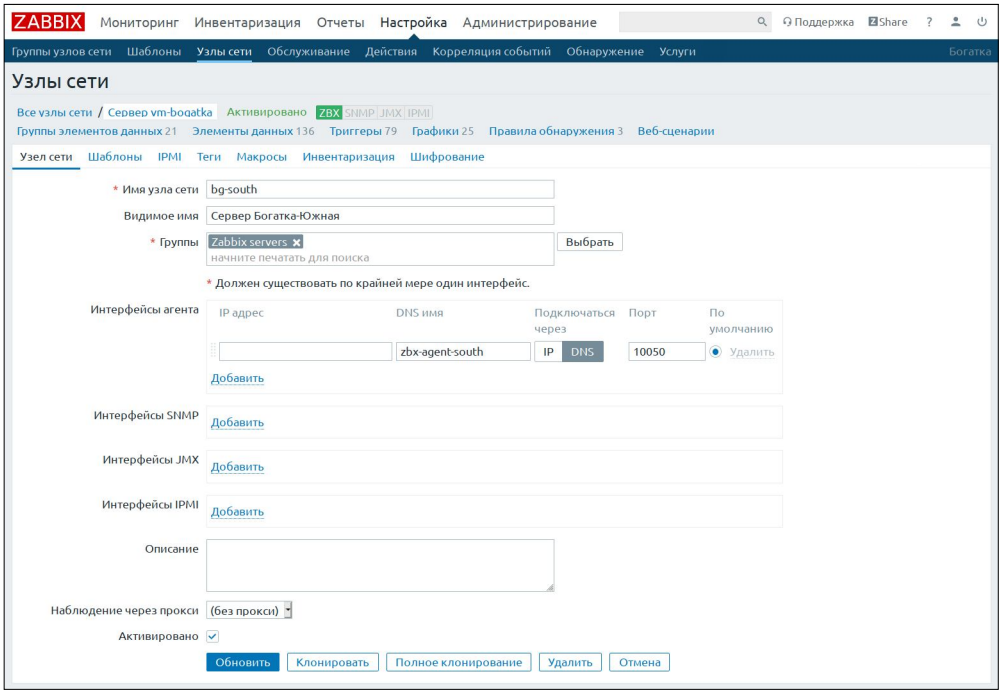
которые отображают некоторые файлы и директории в контейнере на реальные файлы и директории узла. При необходимости список таких директорий может быть дополнен.

Чтобы запустить контейнер "Богатка", следует выполнить скрипт:

```
$ ./bogatka-run
```

§2.5. Установка агента Zabbix

Агент Zabbix — отдельная служба, предназначенная для контроля за состоянием самого физического хоста средствами Zabbix и отображения его на том же сервере Zabbix. Рекомендуется иметь его на каждом физическом хосте. Непосредственно для работы системы "Богатка" он не требуется. Агент работает в отдельном контейнере.



Для его установки следует выполнить скрипт:

```
$ ./zabbix-agent-run
```

и несколько завершающих действий на сервере Zabbix: удалить узел сети Zabbix server, создаваемый по умолчанию, и вместо него создать узлы для каждого из серверов Богатка, имеющихся в системе. В качестве интерфейса для них следует указывать *hostname* контейнера, в котором работает агент на данном хосте, и поиск по DNS; имя задаётся в вышеуказанном скрипте, или по умолчанию формируется в виде zbx-agent-XOCT, где XOCT — это *hostname* хоста. Шаблон для узла см. в файле zabbix-setup.zip .

Если в тестовой системе существует единственный сервер Богатка, то для него можно использовать готовый узел Zabbix server.

§2.6. Сторонние сервисы

Для работы системы необходим или рекомендуется доступ к следующим сервисам, не входящим непосредственно в её состав.

Сервер NTP. Синхронизация времени на всех узлах критически необходима для работы как роя Docker (по спецификации Docker, не допускается рассогласование более 1 сек.), так и непосредственно системы "Богатка". В отсутствие синхронизации вероятны непредсказуемые нежелательные эффекты, такие как отрицательные таймауты, бесконечные времена ожидания ответов и т.п.

Для синхронизации времени на всех узлах должна быть настроена синхронизация времени от одного и того же источника, одним из двух способов:

— Открыть доступ в сети общего пользования по протоколу NTP (UDP, порты источника и назначения 123). Все дистрибутивы Linux включают в себя клиента NTP, настроенного на использование серверов NTP общего пользования.

— Организовать в закрытой корпоративной сети заказчика собственный сервер NTP и перенастроить клиентов NTP на узлах для работы с ним.

Подробно о настройке клиентов NTP см. *man pages* по клиенту, входящему в используемый дистрибутив.

Репозиторий Linux для используемого дистрибутива (или одно из его зеркал) — необходим для обновления ПО Linux на узлах, в т.ч. для получения обновлений безопасности. Следует обеспечить доступ к официальным репозиториям поставщика дистрибутива по протоколам HTTP/HTTPS, FTP, rsync и др. в зависимости от выбранного дистрибутива и репозитория. Если корпоративная политика безопасности категорически не допускает обновление ПО из репозиториев общего пользования, то необходимо предусмотреть процедуру обновления ПО иными способами.

Репозитории Github (*https://github.com/*) и **DockerHub** (*https://hub.docker.com/*) — для получения новых административных скриптов и образов контейнеров. Если доступ к этим репозиториям не допускается корпоративной политикой безопасности, то необходимо предусмотреть процедуру размещения новых образов контейнеров в системе вручную и соответствующим образом отредактировать скрипты.

Сервер геоинформационной системы (ГИС) — необходим для отображения местонахождения клиентов на карте и для взаимного преобразования широты/долготы в адрес, точку на карте и т.п. (см. §3.7, §4.3) В системе предусмотрена возможность использования нескольких некоммерческих серверов ГИС общего пользования.

§2.7. Ресурс HTTP для обновления ПО на клиентах

Поскольку сама система работает по HTTPS, то для раздачи обновлений ПО клиентам удобно использовать встроенный сервер HTTP/HTTPS. В качестве репозитория достаточно использовать один из ведущих хостов системы, имеющий достаточно места на HDD для хранения образов ПО. Следует убедиться, что в скрипте run-bogatka на этом хосте имеется строка:

```
-v /home/bogatka/opt/download:/opt/app/download \
```

Файлы, помещённые в указанную директорию на хосте, будут доступны клиентам по http://*адрес.хоста.в.корпоративной.сети*/download/*ФАЙЛ* . Они также доступны и по HTTPS (поскольку этот порт также проброшен в контейнер), при условии, что сертификат сервера признаётся клиентом или не проверяется.

Внутри данной директории можно организовать структуру поддиректорий, включая символические ссылки, по усмотрению администратора. Например:

```
download
├releases
│ ├acari-client
│ ├bogatka-docker-images
│ └nsg-client
│   ├2.1.A.B
│   ├2.1.X.Y
│   ├latest --> ./2.1.X.Y
│   └stable --> ./2.1.A.B
└uploads
```

§2.8. Перезапуск и обновление контейнеров

Для ручной остановки и запуска контейнеров следует пользоваться штатными командами Docker, например:

```
cd ~/opt/bogatka-docker
docker stop north
docker rm north
./bogatka-run
```

Чтобы обновить контейнер Богатка, следует отредактировать параметр VER в скрипте в соответствии с до новой версией и перезапустить контейнер. Docker загрузит новую версию контейнера с DockerHub и запустит её.

Для систем, не имеющих доступа в Интернет, новые версии контенеров Богатка поставляются в виде файла-архива. Необходимо поместить этот архив на сервер и выполнить команду, например

```
gunzip -c bogatka-image-1.3.11.tar.gz | docker load
```

Чтобы убедиться, что образ загрузился, выполнить команду:

```
docker images
```

В списке должна быть строка с номером новой версии:

```
nsgru/bogatka                1.3.11          ff885a86da6b   10 months ago   113MB
```

В обоих случаях, единожды обновлённый контейнер сохраняется в системе Docker, и впредь загружать его снова не требуется.

ПРИМЕЧАНИЕ Сохранить образ контейнера в виде архива всегда можно на сервере, где этот образ уже установлен, при помощи команды:

```
docker save nsgru/bogatka:1.3.11 | gzip > bogatka-image-1.3.11.tar.gz
```

Если приложение содержит несколько взаимосвязанных контейнеров, то вместо команды docker rm удобнее использовать docker container prune — она удаляет сразу все остановленные контейнеры. Например, для сервера Zabbix скрипт zabbix-run запускает 3 контейнера.

1. Вспомнить имена всех запущенных контейнеров:

```
$ docker ps --format {{.Names}}
```

```
zabbix-web-nginx-pgsql
zabbix-server-pgsql
zabbix-postgres-server
bogatka-db-north
north
zbx-agent-north
```

2. Остановить три контейнера, относящиеся к серверу Zabbix:

   $ docker stop zabbix-web-nginx-pgsql zabbix-server-pgsql zabbix-postgres-server

   zabbix-web-nginx-pgsql
   zabbix-server-pgsql
   zabbix-postgres-server

3. Удалить их:

   $ docker container prune

   WARNING! This will remove all stopped containers.
   Are you sure you want to continue? [y/N] y
   Deleted Containers:
   .....

4. Снова запустить Zabbix:

   $ ./zabbix-run

Особая задача — перезапуск контейнера БД (а также добавление новых контейнеров БД) в процессе эксплуатации системы. Особенность состоит в том, что за время работы статус главной копии БД мог перейти от той, которая была указана как PRIMARY при первоначальной установке, к какой-либо другой. Необходимо просмотреть список БД на любом из работающих хостов (но не на том, на котором БД в данный момент остановлена):

./db-show

определить текущую главную копию, и указать её как PRIMARY в скрипте db-run на рестартуемом или добавляемом хосте.

Это категорически необходимо сделать для той базы, которая была указана главной ранее, поскольку теперь она заведомо не является таковой; если оставить PRIMARY указывающим на неё саму, то она начнёт рассматривать самоё себя в качестве нового кластера из себя одной и не сможет подключиться к работающему кластеру.

§2.9. Выключение сервера

Сервер Богатка предназначен для непрерывной круглосуточной работы в течение, в идеале, всего срока эксплуатации. На случай сбоев электропитания, автоматическое включение сервера должно обеспечиваться аппаратным стартёром или настройкой в BIOS, или тем и другим. Это, однако, создаёт проблему, если сервер необходимо намеренно выключить, например, для переноса на другое место: после команды shutdown он немедленно включается снова. Выключать же сервер просто отключением питания нельзя, поскольку на нём в этот момент могут быть открытые файлы, и файловая система может быть повреждена.

Корректная процедура выключения сервера следующая:

1. Войти на хост по *ssh* как пользователь bogatka .
2. Набрать в командной строке команду

   $ sudo shutdown -P 0 (для Ubuntu)

3. Приготовиться отключить электропитание сервера выключателем или в розетке.
4. Нажать клавишу Enter и как только сервер физически остановится (погаснут индикаторы, остановятся вентиляторы) — отключить питание. Продолжительность паузы, в течение которой можно это сделать — 3–5 сек, в зависимости от используемого способа автоматического включения.

§3. Администрирование системы

§3.1. Вход в систему

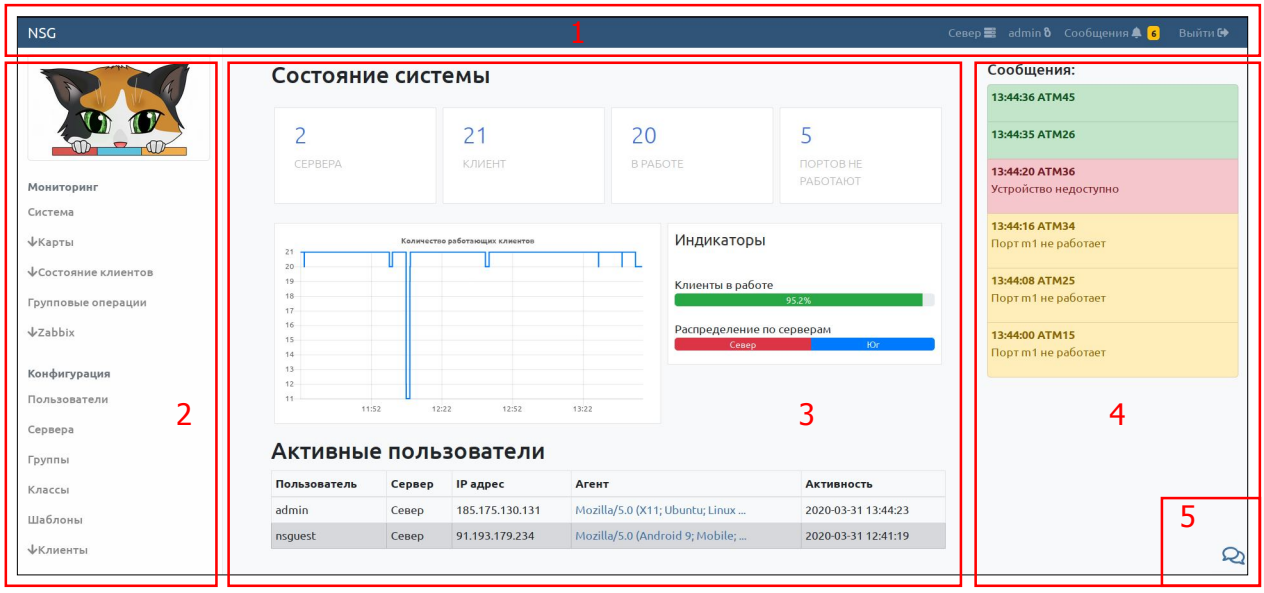
Доступ и работа с системой производится по HTTPS. Для входа в систему необходимо подключиться к серверу "Богатка" по IP-адресу или доменному имени, назначенному одному из узлов роя, и порту TCP, пробрасываемому на порт HTTPS контейнера.

Для работы пригоден любой современный Web-броузер с поддержкой Javascript, в т.ч. броузеры мобильных устройств. Если при установке системы использован самоподписанный или не соответствующий узлу сертификат, необходимо подтвердить использование этого сертификата и запомнить его в броузере. Для первого входа в систему используется имя пользователя admin с паролем admin, установленным по умолчанию.

ВНИМАНИЕ Пароль необходимо немедленно поменять на стойкий случайный пароль, соответствующий принятой корпоративной политике безопасности, особенно, если сервер "Богатка" доступен из сетей общего пользования по HTTPS.

§3.2. Обзор интерфейса

Web-интерфейс системы состоит из следующих областей:



1. Заголовок. В заголовке выводится имя текущего пользователя, имя сервера, на котором он работает, переключатель сообщений и их число, команда для выхода из системы. Выпадающий список рядом с именем пользователя содержит команду для изменения пароля.
2. Панель меню. Условно разделяется на 3 части: "Мониторинг и управление", "Конфигурирование системы", и "Разное".
3. Основная панель. В этой области выводится информация и выполняются действия по существу выбранного пункта меню.
4. Панель сообщений. Уведомления о неработающих компонентах туннелей выводятся в броузер по технологии *push* и соответствуют фактическому состоянию системы на текущий момент.
5. Значок быстрого доступа в чат (см. §5.1) для оперативного обмена информацией между пользователями. Значок зафиксирован в правом нижнем углу и показывается поверх остальных элементов интерфейса.

Цветовая индикация сообщений в панели 4 соответствует степени серьёзности отказа:

— Зелёный — сообщение о восстановлении полноценной работы клиента. Выводится не менее 1 минуты, потом удаляется автоматически.

— Синий — у клиента не работает одно TCP-соединение с одним сервером. Это "штатная нештатная ситуация". Исправляется автоматически средствами системы, если только не является первым проявлением более серьёзного отказа.

— Жёлтый — у клиента не работают все соединения через одного оператора, или все соединения с одним или более (но не всеми) серверами "Богатка". Если проблема не сводится к рутинному пересоединению порта (1–2 мин), а сохраняется в течение длительного времени, это свидетельствует о наличии проблем с подключением на стороне клиента или на стороне серверной площадки, соответственно. Причина может быть как на стороне используемого оборудования, так и на стороне оператора, в т.ч. обрыв кабеля, хищение антенны, исчерпание средств на счету, и т.п.

— Красный — клиент недоступен ни для одного сервера системы, ни по одному каналу связи.

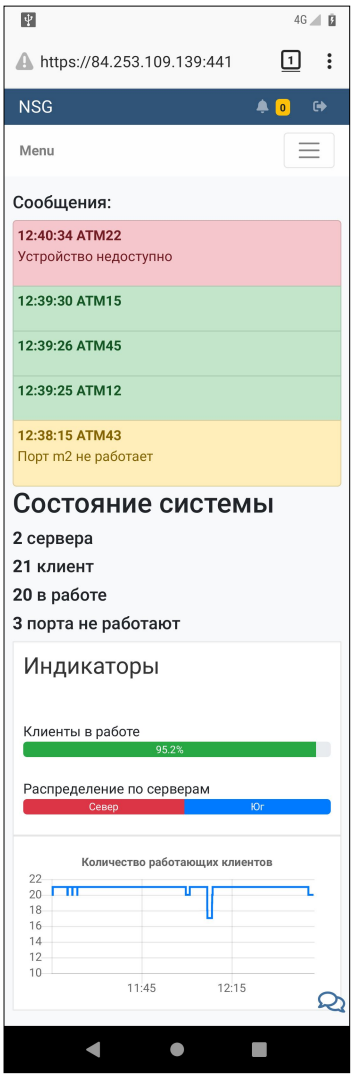
Предназначение системы состоит, прежде всего, в том, чтобы обеспечивать 100%-ю доступность всех включённых в данный момент клиентов. Визуально это означает, что повседневная работа её операторов и полевых инженеров состоит в том, чтобы в первую очередь устранить все "красные" состояния клиентов. Во вторую, по мере возможности, все постоянные "жёлтые" — они являются предвестниками будущих аварий.

При устранении отказа штатными средствами системы сообщение соответственно меняет цвет или удаляется автоматически. Сообщения можно скрыть нажатием переключателя "Сообщения" в заголовке, чтобы расширить основную область. Повторное нажатие включает сообщения снова.

При наведении курсора в область сообщений её изменения приостанавливаются на 5 сек. Этого времени достаточно, чтобы нажать ссылку на интересующего клиента в заранее выбранном сообщении.

На заглавной странице системы выводится сводная информация о компонентах системы: серверах, клиентах (туннелях) — последние 32 точки, пользователях (только для администратора), проблемах и т.п. Web-интерфейс имеет адаптивный дизайн и автоматически модифицируется под имеющийся размер экрана, в т.ч. для экранов мобильных устройств.

Большинство страниц интерфейса содержит подробную информацию по однотипным объектам, представленную в виде таблиц. Для таблиц предусмотрены общепринятые сервисы: постраничный вывод с заданным числом строк на странице, сортировка по любому столбцу в прямом и обратном направлении, поиск по произвольной подстроке, выбор видимых столбцов (для выбора нескольких элементов списка используются Shift-click и Ctrl-click), вывод таблицы в файл формата .csv и на печать.



Главным объективным показателем работы системы является процент доступных клиентов. Для корректного подсчёта этого показателя следует своевременно исключать из него тех клиентов, которые в данный момент заведомо не должны работать, например, находятся на складе или в ремонте. Таких клиентов следует блокировать или перезапускать (см. §3.7), в зависимости от принятого регламента работы.

Для выхода из системы следует нажать кнопку "Выход" и подтвердить действие, или закрыть вкладку броузера или сам броузер, или выключить компьютер. Если это не сделано (компьютер оставлен в "спящем" состоянии, потеряна связь с мобильным пользователем и т.п.), пользователь будет считаться присутствующим в системе.

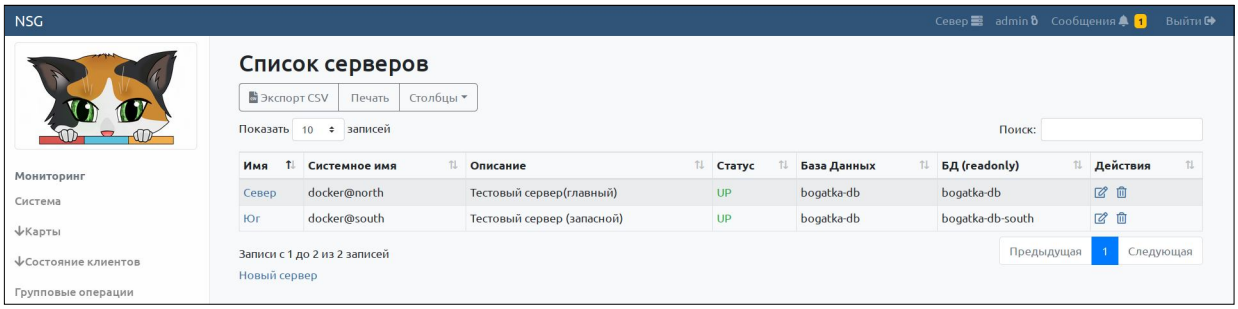
Если пользователь не проявляет активности более 1 часа, он подсвечивается светло-серым цветом, более 24 часов — тёмно-серым.

Все настройки системы, сделанные пользователем, сохраняются в самой системе и доступны из любого броузера, с любого хоста. Непосредственно в броузере сохраняются только токены авторизации и настройки просмотра страниц.

ПРИМЕЧАНИЕ Система находится в состоянии развития, поэтому фактический вид интерфейса может отличаться от приведённого на рисунках в данном документе.

§3.3. Настройка серверов и общие правила настройки

Для настройки серверов следует выбрать в меню пункт "Сервера". На примере данного раздела будут показаны также некоторые общие моменты, относящиеся и к конфигурации других объектов.

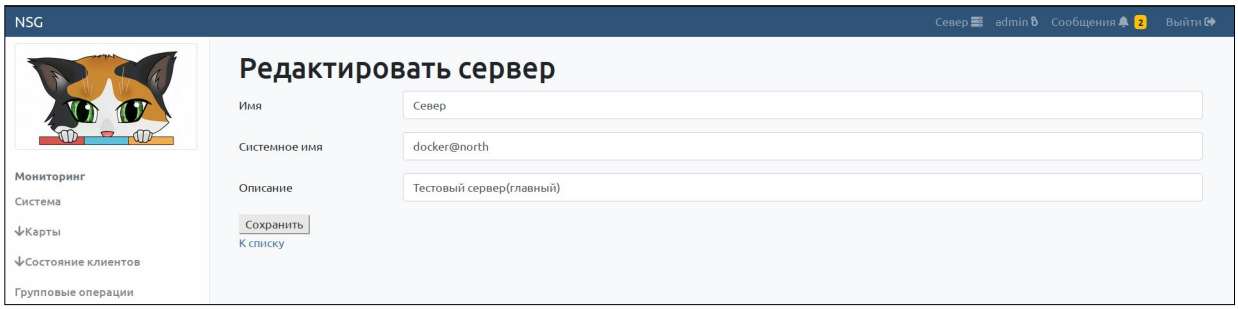


Список серверов, известных системе, представлен в виде таблицы. Для каждого элемента в последнем столбце таблицы (он не выводится на печать и в .csv) содержатся кнопки действий с ним: редактировать, удалить и др. в зависимости от его типа. Имя элемента (в данном случае, сервера) является ссылкой на следующую страницу со свойствами данного элемента:

Кнопка "Редактировать" в обоих случаях открывает страницу для изменения настроек сервера:

В данном случае, свойства сервера включают:

Имя — Произвольное имя, под которым данный сервер будет фигурировать в пользовательском интерфейсе системы.



Системное имя
: Формируется из константы docker@ и *hostname* сервера; последнее совпадает с именем контейнера (параметр -n в настройках контейнера, см. §2.4.). Это ключевой параметр, по которому сервер идентифицируется в системе.

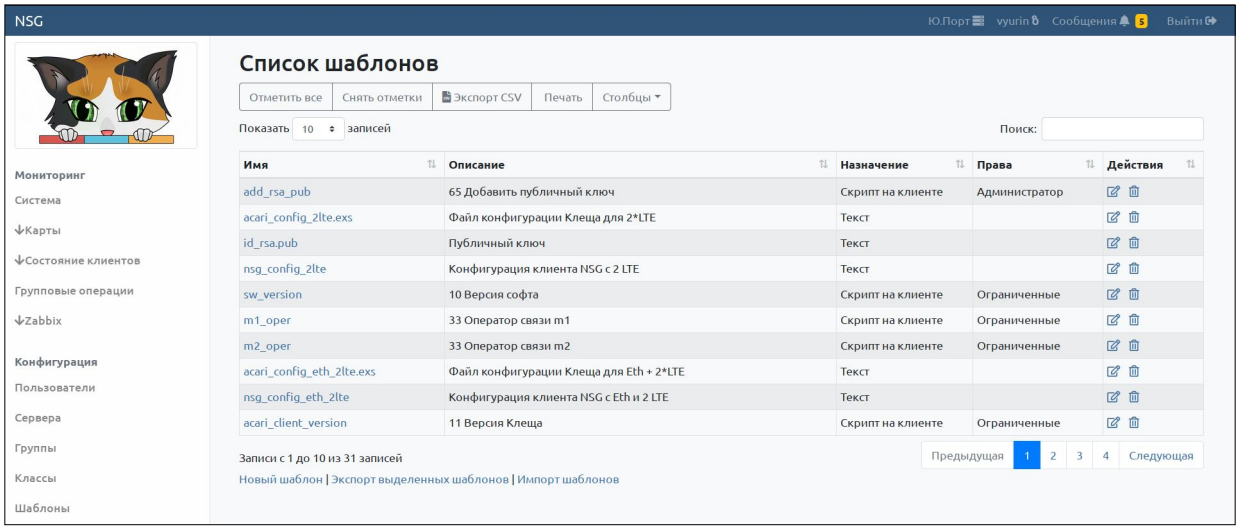
Описание
: Произвольное текстовое описание для удобства администрирования, на работу системы не влияет.

ВНИМАНИЕ На всех страницах редактирования после завершения работы необходимо нажать кнопку "Сохранить".

Первоначально каждому серверу системы известен только он сам. Для добавления других серверов следует использовать кнопку "Новый сервер" на сводной странице. Она открывает страницу редактирования, но с пустыми полями. (Аналогично в других разделах конфигурации; некоторые значения могут вычисляться автоматически или подставляться по умолчанию). После добавления новый сервер становится полноценным узлом системы, и на него могут направляться клиенты. Дальнейшую настройку (в т.ч. добавление очередного нового сервера) можно выполнять на любом из серверов.

§3.4. Настройка шаблонов

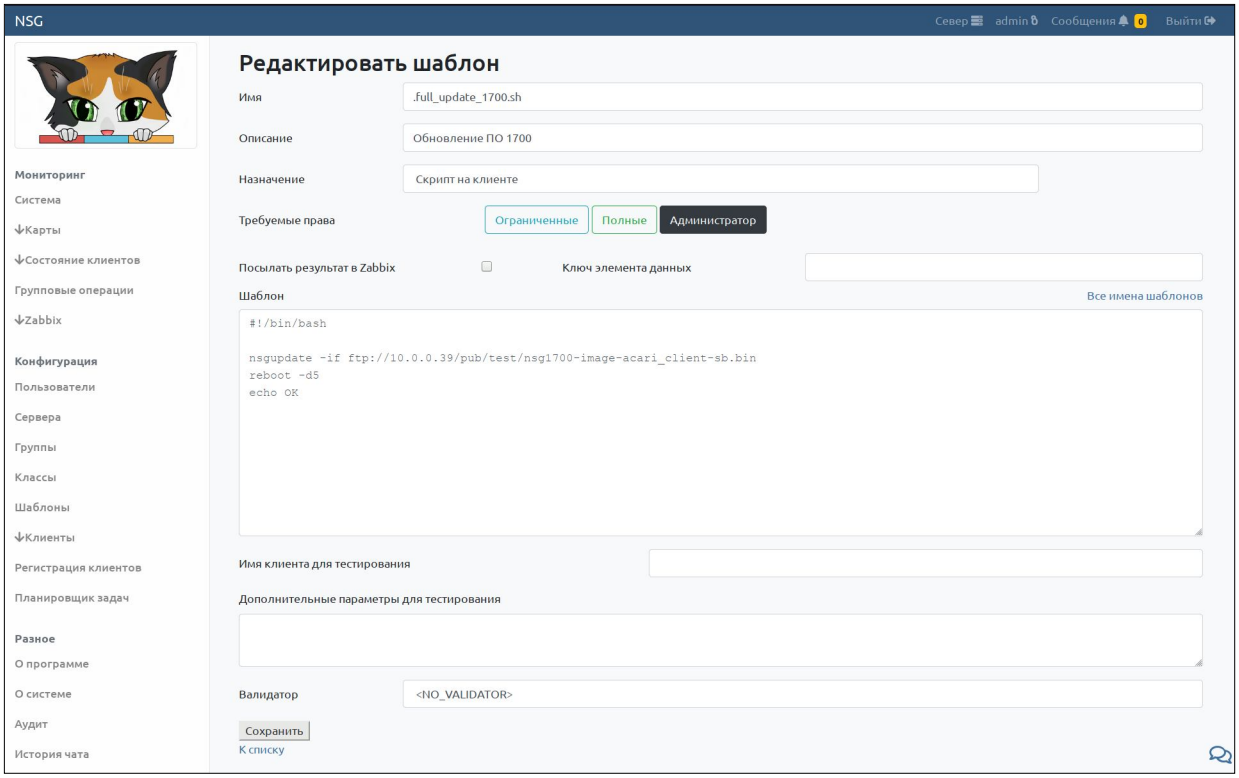
Настройка, управление и мониторинг системы состоят, в конечном счёте, из передачи и применения текстовых файлов (*скриптов*) на объекты управления: клиентские устройства и сами сервера "Богатка".



Первичным элементом управления в системе являются *шаблоны* скриптов. Шаблон — это текстовый файл, абстрагированный от индивидуальных параметров каждого конкретного клиента. Шаблон составляется администратором на основе эталонной конфигурации, отработанной при настройке первого пробного туннеля вручную. Уникальные параметры клиента заменяются *макроподстановками*.

Некоторые макроподстановки являются встроенными и вычисляются автоматически из идентификатора клиента (и, возможно, другой информации, которую он передаёт о себе при подключении к системе). Все остальные подстановки, использованные в шаблоне, рассматриваются как параметры, требующие ручного ввода или выбора (в зависимости от использованного формата); для них создаются соответствующие поля на страницах редактирования клиентов (см. §3.5, §3.7).

В результате применения макроподстановок для конкретного клиента из шаблона генерируется уникальный скрипт, содержащий индивидуальные параметры этого клиента. Как частный случай, шаблон может не содержать макроподстановок вообще; такой шаблон тождественен скрипту и одинаков для всех клиентов.



На страницах редактирования и просмотра свойств шаблона имеются следующие атрибуты:

Имя Имя шаблона в системе. Может содержать буквы, цифры, точку и знак подчёркивания (_). Другие спецсимволы запрещены во избежание проблем их интерпретации в разных командных оболочках и языках. Расширения (.exs, .sh и др.) не являются обязательными и служат лишь для подсказки относительно предназначения того или иного шаблона.

Специальными являются имена, начинающиеся с точки. Такие имена следует использовать для критически ответственных и потенциально опасных скриптов, таких как обновление ПО или изменение конфигурации клиента. Скрипты с такими именами не показываются на странице групповых операций (см. §4.5), если не установлено явно опция "Разрешить опасные скрипты".

Можно начинать имя шаблона с цифрового префикса (например, двузначного — категория и номер в этой категории), чтобы упростить классификацию скриптов и их сортировку в списках.

Имя шаблона не может быть изменено впоследствии. Если нужно изменить имя, то скрипт необходимо клонировать с новым именем, а затем вручную проверить и исправить ссылки на него, которые могут быть в других скриптах.

Описание Текстовое описание шаблона, для понимания выполняемых им действий. Непосредственного влияния на работу шаблона не оказывает.

Назначение

Предназначение данного шаблона. В зависимости от него, шаблоны и скрипты будут показываться или скрываться при выводе на различных страницах, в различных полях и т.п., чтобы не перегружать список излишними сущностями. Возможные значения:

Скрипт на клиенте
: Исполняемый скрипт, предназначенный для выполнения на клиентских устройствах определённого класса.

ПРИМЕЧАНИЕ Для устройств NSG командной оболочкой в рамках данной системы является *bash*. Фирменное командное дерево NSG доступно посредством вызова оболочки *nsgsh* из скриптов *bash*.

Скрипт на сервере
: Исполняемый скрипт, предназначенный для выполнения на серверной стороне туннелей (для клиентов из определённого класса).

В обоих случаях исполняемый скрипт должен содержать корректную последовательность команд CLI для той модели оборудования, для которой он предназначен.

Zabbix API Скрипт для обработки информации, хранящейся на сервере Zabbix. Запрашивает у Zabbix указанные данные для указанного туннеля с помощью WebAPI Zabbix и обрабатывает их указанным образом. Предназначен для случаев, когда требуемую информацию проблематично получить в самом Zabbix, или когда её необходимо представить именно средствами системы "Богатка". Исполняется на серверах "Богатка". Подробнее см. §6.11.

Текст Произвольный текстовый файл, не предназначенный для исполнения где-либо. Это может быть, в частности, файл конфигурации "Клеща" на клиенте, или какого-либо иного приложения, или ключ. Единственное, что можно с ним сделать — это с помощью отдельного исполняемого скрипта передать его на устройство и там поместить в предусмотренное для него место. Синтаксис неисполняемого скрипта определяется тем приложением, для которого он предназначен.

Вставка Текстовый файл, используемый не самостоятельно, а как часть другого шаблона (исполняемого или неисполняемого). В отличие от текста, содержит, по существу своему, переменные окружения или собственные макроподстановки, значения которых определяются в вызывающем скрипте. Примеры использования см. в §6.9.

Терминал Дополнительное терминальное окно на странице мониторинга клиента (см. §4.4). Тело шаблона содержит команду, которая будет автоматически выполнена (в контексте *bash*) после подключения по SSH. Например:

`nsgsh` — будет запущена командная оболочка NSG

`nsgcu port.a1 -s 9600 -t 8n1 --nohwfc` — будет запущена консольная утилита на порту `a1`.

На странице мониторинга клиентов в тех классах, к которым применяется данный шаблон, выводится дополнительная кнопка. В качестве названия кнопки используется описание шаблона, например:

`Порт a1 (9600)`

Как и во всех шаблонах, для терминалов допускается использовать макроподстановки. Например, скорость в последнем примере можно описать в качестве параметра и устанавливать её индивидуально на каждом клиенте в зависимости от того, что и с какой скоростью консольного порта подключено к нему на данной площадке.

Системный
: Шаблон, вызываемый средствами системы при определённых обстоятельствах. Не предназначен для вызова или редактирования пользователями.

Нет
: Категория шаблона не определена.

Требуемые права
: Только для исполняемых скриптов: права, которые должен будет иметь пользователь, чтобы выполнить данный скрипт. Подробно о разделении прав между пользователями, действиями и группами см. §3.9.

— Ограниченные — Скрипт доступен для всех пользователей, имеющих доступ к данной группе клиентов.

— Полные — Скрипт доступен только для пользователей, имеющих полный доступ к данной группе.

— Администратор — Скрипт доступен только для администраторов системы.

Выгрузка
: Опция: включить данный скрипт в список доступных на странице выгрузки (см. §5.5). По существу её следует использовать для скриптов, суть которых не в том, чтобы выполнить команду (например, "Рестарт клиента"), а в том, чтобы извлечь из клиента некоторую информацию. Информация от последнего выполнения скрипта сохраняется в самой системе "Богатка"; для скриптов, выполняемых регулярно и многократно, следует использовать передачу этой информации на сервер Zabbix.

Ключ элемента данных
: Имя параметра, которое будет передаваться в Zabbix. Результат выполнения скрипта передаётся в виде ключ=значение, например, operator[m1]=mts . Если данное поле пустое, то информация в Zabbix не передаётся.

Шаблон
: Собственно тело шаблона, содержащее макроподстановки. Подробно о написании шаблонов и языке макроподстановок см. §6.1.

Результатом макроподстановки всегда является текстовая строка, которая вставляется в тело шаблона "как есть", без кавычек. Если синтаксис команд и конфигурации для конкретного оборудования требует наличия кавычек или, наоборот, их отсутствия, то это необходимо контролировать вручную при разработке шаблона.

Ссылка "Все имена шаблонов" позволяет просмотреть имена шаблонов, определённых в системе на данный момент, чтобы включить в тело редактируемого шаблона корректную ссылку на какой-либо другой.

Имя клиента для тестирования
: Тестовый идентификатор клиента (в формате МОДЕЛЬ_ЗАВ.НОМЕР, например, NSG1700_2001012345) для генерации тестового скрипта.

Дополнительные параметры для тестирования
: Другие входные данные для генерации тестового скрипта — уникальные параметры клиента, которые должны быть известны на момент тестирования. Вводятся в формате JSON { " ключ": "значение", ... }, где каждый ключ — одна из подстановок, указанных в теле шаблона.

Полученный тестовый скрипт выводится на странице свойств шаблона.

Валидатор
: Имя утилиты для автоматической проверки корректности получаемого скрипта (необязательное). Валидирование (если валидатор назначен) производится при нажатии кнопки "Сохранить". Если скрипт содержит ошибку, то будет выведено сообщение с указанием проблемного места (но шаблон, тем не менее, будет сохранён).

В отсутствие валидатора скрипт можно проверить вручную на странице свойств шаблона.

Страница свойств шаблона содержит, наряду со стандартными ссылками, две специфические:

Клонировать
: Создать новый шаблон на основе текущего. То же, что и "Новый шаблон", но все поля изначально заполняются так же, как в данном шаблоне. Некоторые из них необходимо изменить, в т.ч. имя шаблона — обязательно.

История изменений
: Открывает отдельную страницу с историей редактирования данного шаблона. При необходимости она позволяет выяснить, кто из пользователей вносил изменения в шаблон, какие именно, и откатить изменения к одной из предыдущих версий.

Кнопка (i) на странице просмотра свойств шаблона выводит список всех данных, которые доступны в этом шаблоне.

На общей странице Шаблоны, помимо повсеместной кнопки Новый шаблон, имеются две специфические:

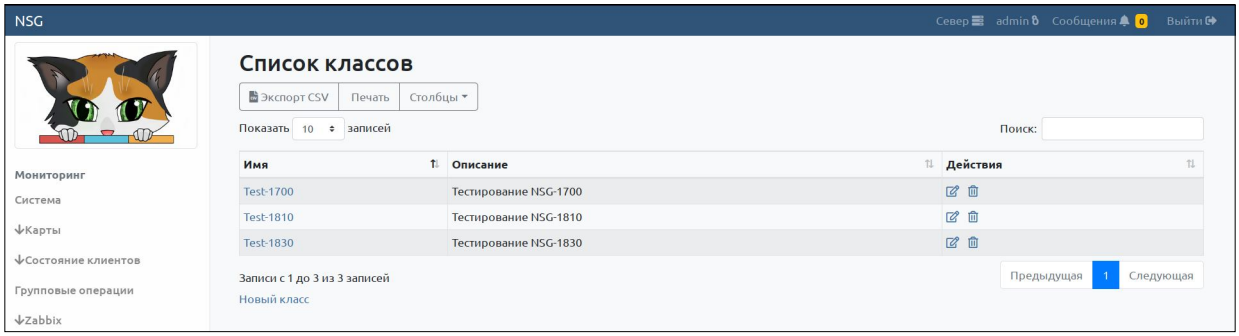
Экспорт выделенных шаблонов
: Экспортировать шаблоны в файл формата JSON для создания резервной копии, переноса на другую инсталляцию и т.п. Для выбора шаблонов, подлежащих экспорту, используются стандартные операции click, Shift-Click, Ctrl-click.

Импорт шаблонов
: Импортировать готовый набор шаблонов из файла.

§3.5. Настройка классов

Классы позволяют структурировать всё множество клиентов системы с точки зрения настройки. *Класс — это совокупность клиентов, к которым применяются одни и те же шаблоны и скрипты*. Как правило, к одному классу целесообразно отнести всех клиентов одной и той же модели, с одинаковой аппаратной комплектацией и схемой подключения. Но это требование не жёсткое, например, в одном классе могут быть (хотя это не всегда удобно для работы) устройства нескольких моделей и модификаций, к которым подходит одна и та же конфигурация.

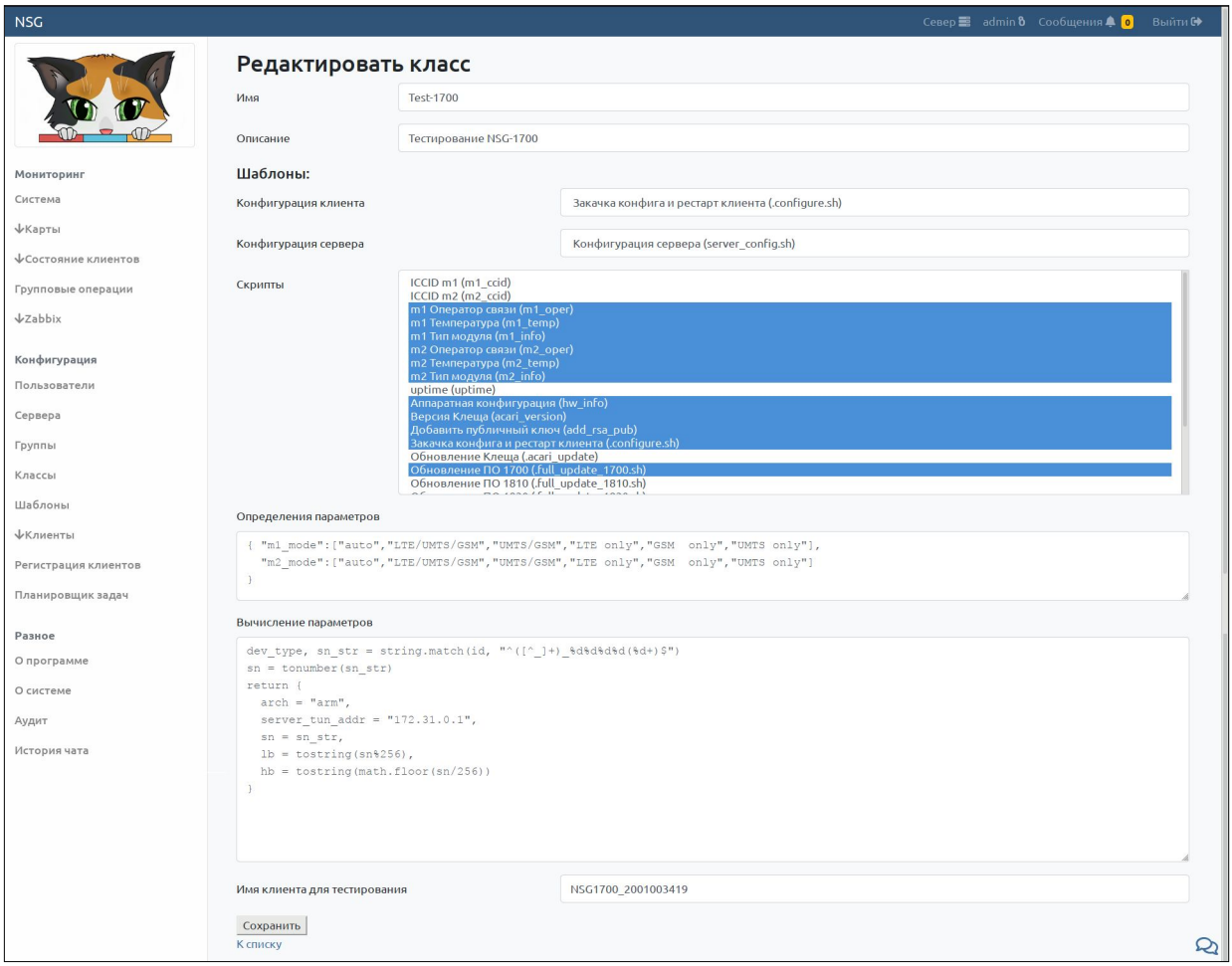
Каждый клиент в системе может принадлежать к одному и только к одному классу, поскольку конфигурация в него заливается ровно одна. Клиент, не включённый ни в какой класс, может числиться в системе лишь номинально, но он не сможет получить рабочую конфигурацию (ибо неизвестно, какая конфигурация ему положена) и, соответственно, не сможет работать.



На страницах просмотра и редактирования свойств класса имеются следующие атрибуты:

Имя
: Произвольное имя для идентификации класса в системе.

Описание
: Текстовое описание данного класса для удобства администрирования. Непосредственно на работу системы не влияет.



Конфигурация клиента
: Исполняемый шаблон для настройки клиента в системе.

Конфигурация сервера
: Исполняемый шаблон для настройки серверной стороны туннеля (если требуется).

Для всех клиентов одного класса используется всегда один и тот же шаблон. То же самое относится к серверной стороне. В этом и состоит сущность класса.

Скрипты
: Список скриптов, которые могут быть применены к туннелям данного класса. В списке показываются все исполняемые скрипты, определённые в системе (как для клиентской, так и для серверной стороны — выбор, на какой стороне исполняется скрипт, будет сделан при его запуске, см. §4.5). Для выбора нескольких элементов списка используются Shift-click и Ctrl-click .

Определения параметров
: Определения всех макроподстановок, использованных в выбранных шаблонах и задаваемых вручную. В зависимости от этих определений формируются поля для ввода или выбора параметров на странице редактирования клиентов данного класса (см. §3.7). Для описания используется формат JSON. Пример:

```
{  "myfreemacro" :"defaultvalue",
   "myDefaultEmptyMacro" :"",
   "mylistmacro" :["defaultvalue", "value1", "value2", ...],
   ...
}
```

Здесь первая макроподстановка предполагает свободный ввод в виде текстовой строки. Вторая — то же самое, но значение по умолчанию пустое. Третья — выбор из заданного списка; первым указывается значение, устанавливаемое по умолчанию.

Вычисление параметров

Правила расчёта параметров, вычисляемых автоматически. На практике исходной информацией для расчёта является идентификатор клиента, состоящий из модели и порядкового серийного номера устройства (последние 6 цифр; первые 4 цифры заводского номера означают год и месяц выпуска). Из него формируются, в показанном выше примере, три строковые переменные: порядковый номер в десятичном формате, а также его старший и младший байты (предполагается, что номер не превышает 65535) в шестнадцатеричном формате — их удобно использовать для автоматического назначения IP-адресов. Некоторые параметры могут назначаться фиксированным образом для данного класса, например, для устройств NSG–1700 архитектура процессора равна arm .

Правила расчёта описываются на языке Lua 5.2 без использования небезопасных библиотек (*os*, *file*, *io*). Минимальные сведения о Lua см. в §4.6.

Как частный случай, допускается определение вычисляемого параметра из вводимых вручную, например:

```
b3 = client.params.my_manual_byte3,
b4 = client.params.my_manual_byte4,
ip_addr = "172.16.b3.b4"
```

Имя клиента для тестирования

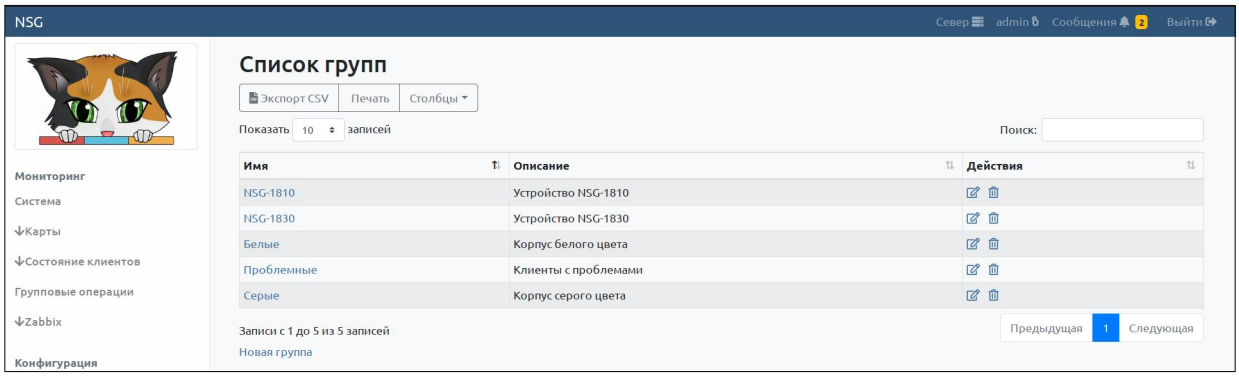
Тестовый идентификатор клиента (в формате МОДЕЛЬ_ЗАВ.НОМЕР, например, NSG1700_2001012345) для проверки правильности вычисления параметров. Просмотреть результат для этого клиента можно на странице свойств класса.

На странице свойств класса имеется специфическая ссылка Клонировать для создания нового класса на основе заданного.

ПРИМЕЧАНИЕ При переносе клиента в другой класс рекомендуется выполнить для него сброс в заводскую конфигурацию, чтобы настроить его "с чистого листа". При изменении свойств класса рекомендуется выполнить сброс для всех клиентов, включённых в данный класс.

§3.6. Настройка групп

Группы, в отличие от классов, позволяют структурировать всё множество клиентов системы с точки зрения управления и мониторинга. С их помощью можно разделить клиентов по разным практически значимым критериям, в том числе по территориальному размещению, административному подчинению, степени важности, аппаратной модификации (если она не влечёт различий в конфигурации) и т.п. Это позволяет регулировать видимость клиентов в интерфейсе более или менее обозримым числом — что особенно важно при массовых инсталляциях, в которых клиенты могут исчисляться тысячами и десятками тысяч. Так же ограничиваются и разделяются операции управления, например, обновление ПО производится в первую очередь на группе "подопытные кролики", затем на группе "обычные", и в последнюю очередь, если это ПО проявило себя положительно — на критически важных клиентах группы "VIP".



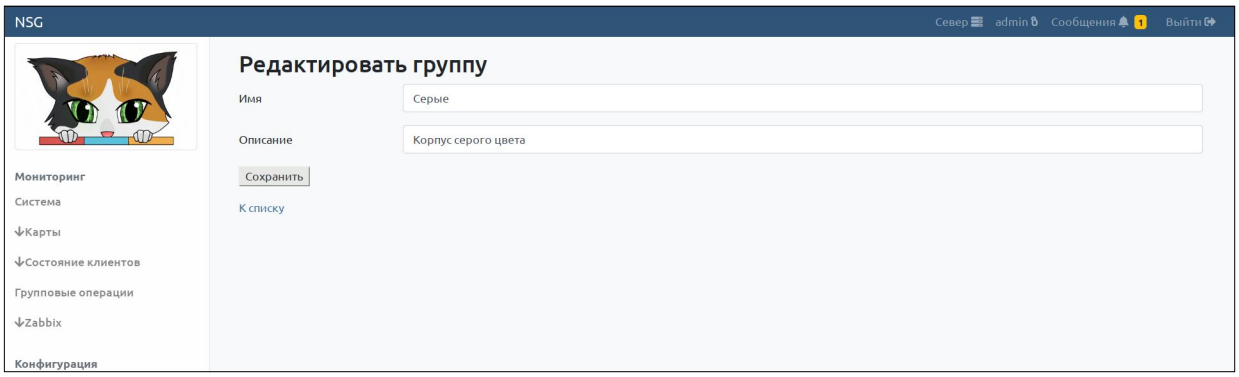
Группы являются ключевым инструментом для разделения полномочий относительно тех или иных клиентов между различными пользователями. Рядовым пользователям системы (не администраторам) назначаются права применительно к каждой конкретной группе: полные (мониторинг и управление), ограниченные (только мониторинг и, возможно, отдельные операции управления) или никакие (в этом случае пользователь вообще не видит этих клиентов в интерфейсе и не имеет доступа к ним). Подробно о настройке пользователей и управлении их правами см. §3.9.

Само по себе членство в группах не оказывает непосредственного влияния на работу клиентов. Один и тот же клиент может, в отличие от классов, быть включённым в несколько разных групп. В частности, рекомендуется использовать группы "Новые" и "Проблемные" для клиентов, требующих особого внимания.

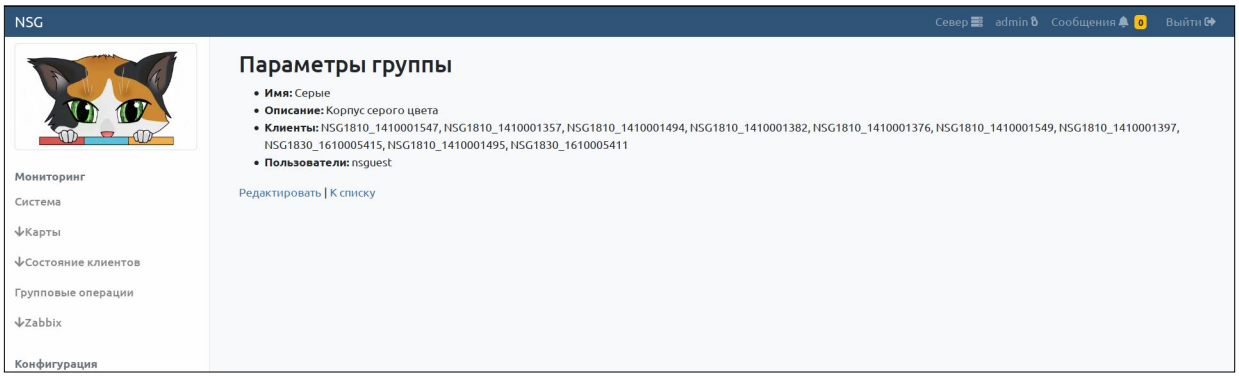
В процессе эксплуатации системы могут создаваться и удаляться новые группы по результатам работы скриптов и фильтров. Такие группы далее в данном документе называются *временными*, и при создании для них автоматически предлагается имя, состоящее из префикса _TMP_ и описания фильтра. Но фактически они ничем не отличаются от остальных, и им можно дать (в момент создания) любое другое имя.

Например, в такую группу можно выделить клиентов, на которых по какой-то причине не было обновлено ПО до последней версии, или на которых используются SIM-карты определённого оператора, выполнить с ними требуемые действия, и затем удалить группу. Подробно о групповых операциях см. §4.5.

На странице редактирования группы определяется только её имя и текстовое описание. Других атрибутов группа не имеет.



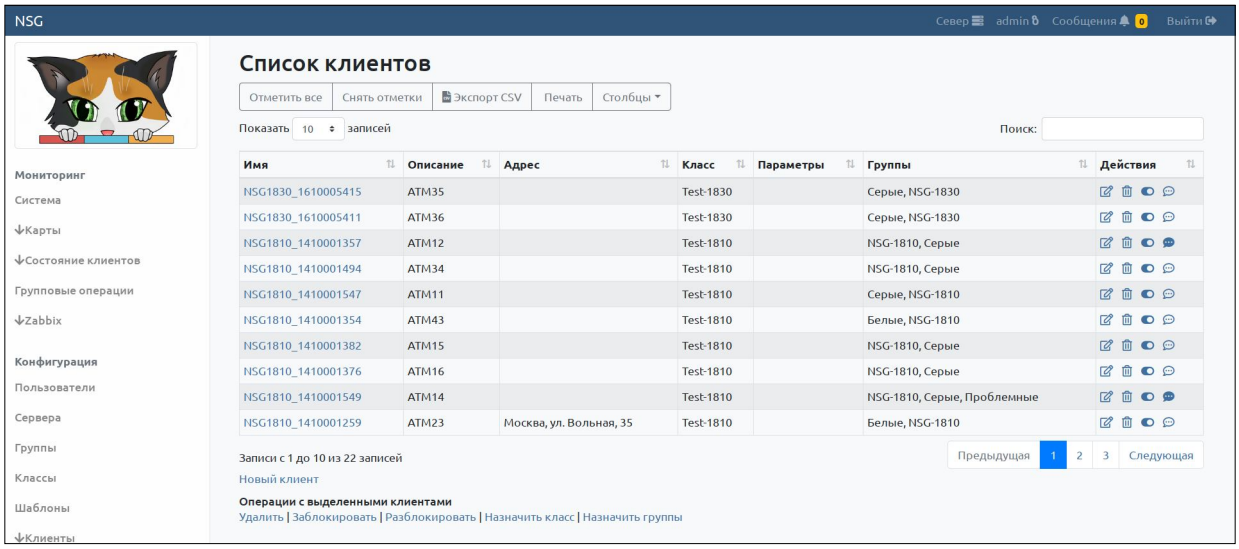
На странице свойств группы можно просмотреть список клиентов, включённых в группу, и список пользователей, имеющих какие-либо права доступа к данной группе. (Не считая администраторов, которые имеют доступ ко всем группам по определению.)



§3.7. Настройка клиентов (туннелей)

Заключительный этап для начала работы системы — это настройка клиентов (туннелей). Она связывает воедино сделанные ранее настройки шаблонов, классов и групп. Подменю "Конфигурация —> Клиенты" позволяет выбрать для работы часть клиентов, которым ранее была назначена определённая группа (из числа групп, доступных данному пользователю), либо "Все" для работы со всеми доступными клиентами.

Страница конфигурации клиентов содержит специфические операции для каждого клиента и для нескольких отмеченных клиентов. Отметить клиентов можно с помощью стандартных действий click, Shift-click, Ctrl-click в любом месте строки таблицы (кроме ссылок и кнопок), а также с помощью кнопок Отметить все и Снять отметки. Сделанный выбор относится ко всем страницам таблицы (не только к видимой в данный момент) и сохраняется при переходе на другие страницы.



Заблокировать/Разблокировать

Запретить работу клиента или выделенных клиентов, не удаляя его при этом из системы полностью. Блокировка решает одновременно несколько задач:

— Безопасность. Если есть подозрение, что клиентское устройство похищено злоумышленниками, которые могут попытаться с его помощью проникнуть в корпоративную сеть, то его следует срочно отключить от системы до выяснения обстоятельств.

— Корректный подсчёт статистики. Заблокированные клиенты в подсчёте не учитываются, таким образом, суммарная статистика показывает процент работающих клиентов от числа тех, которые должны работать в данный момент (в идеале 100%, в этом и состоит смысл существования данной системы), а не от всех находящихся "в поле", лежащих на складе и болтающихся неизвестно где, вместе взятых.

— Разгрузка области системных сообщений и страниц мониторинга от клиентов, заведомо выключенных в данный момент. Это не просто удобство работы — это важно, чтобы в потоке ложных сообщений не пропустить аварийное состояние реально работающего клиента. Ибо, как было сказано в одной старой книге, к пастуху, постоянно кричавшему без причины "волки, волки!", никто не пришёл на помощь, когда волки действительно напали на его стадо.

— Разгрузка экрана групповых операций (см. §4.5). Заблокированные клиенты не показываются на данной странице и не участвуют в операциях. В противном случае они будут висеть в списке неограниченно долго в состоянии ожидания ответа, бессмысленно загромождать таблицу и требовать дополнительного фильтра для их отбраковки.

Перезапустить

Заблокировать клиента и разблокировать его обратно. В результате этого действия клиент по-прежнему может работать, но переводится в состояние "ни разу не подключался" (см. §4.2). Это позволяет разгрузить статистику от неактивных клиентов, а экран — от аварийных сообщений, если клиент по той или иной причине временно выведен из эксплуатации.

Комментарий

Произвольный текстовый комментарий, который может быть назначен любому клиенту. Непосредственно на работу клиента не влияет. В комментариях рекомендуется отражать особенности их состояния и настройки, например: "Сеть LTE работает неустойчиво, назначен вручную режим UMTS/GSM". Таким образом, пул клиентов становится самодокументируемым. Это поможет другим пользователям разобраться в ваших действиях в случае вашего отпуска, увольнения или скоропостижной смерти, да и вам самим через неделю уже трудно будет вспомнить, зачем вы сделали ту или иную настройку.

Свои собственные комментарии можно редактировать. Чтобы удалить свой комментарий, надо стереть его и нажать кнопку "Сохранить". Администраторы имеют право удалять также чужие комментарии (например, если проблема устранена).

Вид значка на кнопке изменяется при наличии/отсутствии комментариев.

Удалить

Удалить клиента (или выделенных клиентов) из системы.

Назначить класс (для выделенных клиентов)

Назначить всем отмеченным клиентам новый класс. Изначальный класс у выбранных клиентов может быть разным, он в любом случае игнорируется. Это может потребоваться, например, когда некоторая часть клиентов переводится на другую схему подключения или требует каких-либо специфических настроек.

Назначить группы (для выделенных клиентов)

Назначить всем отмеченным клиентам новый набор групп. Изначальное членство в группах может быть разным для разных клиентов, именно поэтому оно сбрасывается. Это может потребоваться, например, когда некоторая часть клиентов переводится в иное административное подчинение.

QR

Сгенерировать QR-коды для доступа на инженерные страницы клиентов (см. §5.4). Полученные коды можно, например, распечатать на этикетках и наклеить на устройства перед их передачей низовым техникам. Отсканировав код с помощью смартфона, техник попадает на отдельную страницу, где представлена информация о состоянии данного устройства, обнаруженные неполадки и указания искусственного интеллекта Богатки (см. §5.3) по их устранению.

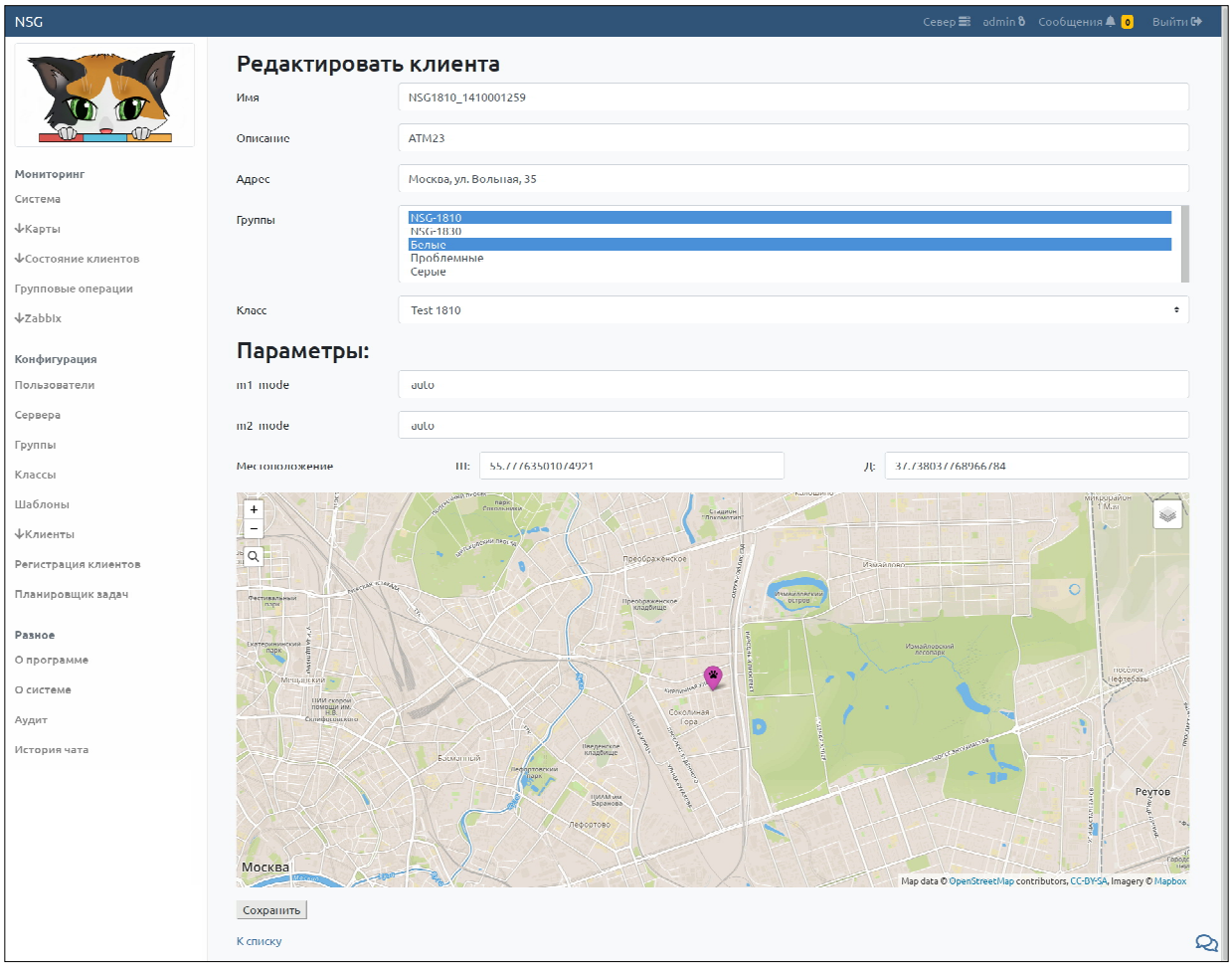
Одновременно QR-код можно использовать в качестве инвентарного номера устройства, чтобы не писать его на корпусе масляной краской, как это было принято испокон веков.

На странице редактирования клиента имеются следующие поля:

Имя

Системный идентификатор клиента. Именно таким образом клиент идентифицирует себя в системе. Для того, чтобы он мог начать работу, он должен быть либо зарегистрирован в системе под этим именем заранее, либо допущен к работе вручную после того, как будет обнаружен системой (см. §3.8).

Имя может быть произвольным, единственное требование — оно должно совпадать с тем, что передаёт клиент в запросе конфигурации. На практике, на момент написания данного документа, оно составляется из наименования модели (без тире) и заводского номера устройства. Последний, в свою очередь, состоит из 4 цифр года и месяца выпуска и 6 цифр порядкового серийного номера.



Следует помнить, что из имени могут вычисляться (и вычисляются на практике) некоторые уникальные параметры конфигурации для данного клиента (см. §3.5). Но это не догма, они могут вводиться и другими способами или передаваться в строке запроса.

ВНИМАНИЕ Имя клиента вводится один раз при его создании, и впоследствии изменяться не может. Если имя введено неправильно, необходимо удалить клиента и создать его заново.

Описание Текстовое описание клиента для удобства администрирования. Непосредственно на работу клиента не влияет. В частности, это может быть системное имя площадки в прикладной системе, например, Подстанция 13 или АТМ–1234567.

Адрес Адрес установки клиента, или любая другая информация для удобства администрирования. Непосредственно на работу клиента не влияет.

Группы Группы, в состав которых включён данный клиент. В списке для выбора показываются все группы, имеющиеся в системе. Для выбора нескольких групп используются Shift-click и Ctrl-click .

Класс Класс, к которому отнесён данный клиент. Непосредственно определяет его конфигурацию и набор других скриптов, доступных для работы с ним.

Параметры

Макроподстановки, упоминаемые в шаблонах для данного класса и определённые для ручного ввода или выбора. Названия и формат полей ввода формируются в соответствии с определениями параметров в свойствах класса (см. §3.5).
Однотипные параметры для нескольких портов (или иных объектов конфигурации) размещаются в несколько колонок. Для этого они должны содержать символ "_" (подчерк) и общий суффикс после него, например: m1_mode, m2_mode.

Местоположение

Географическая широта и долгота точки, в которой находится устройство. Может указываться любым из трёх способов, которые автоматически разрешаются один в другой:

— С помощью мыши на карте.

— Вводом адреса в поле поиска (значок лупы в левом верхнем углу карты).

— Вводом координат в поля Ш и Д . Если клиент оборудован приёмником GPS и передаёт свои координаты в строке запроса конфигурации, то они подставляются автоматически.

Значок "Слои" в правом верхнем углу карты позволяет выбрать источник карт из числа общедоступных серверов, либо Custom для корпоративного или платного сервера ГИС, указанного при запуске контейнера (см. §2.4.).

ПРИМЕЧАНИЕ Для работы карт необходим доступ к публичному или корпоративному серверу ГИС.

ПРИМЕЧАНИЕ Поскольку справочные параметры (описание, адрес, широта/долгота/положение на карте) не влияют на работу системы, можно отдельно разрешить их редактирование для пользователей с ограниченными правами. Это делается на странице системных настроек, см.§3.11.

На странице свойств клиента выводятся все указанные параметры. Страница содержит также кнопки Блокировка, Комментарии и две специфические операции:

Клонировать

Создать нового клиента с аналогичными параметрами (имя и некоторые другие потребуется, конечно, изменить).

Мониторить

Перейти на страницу мониторинга и управления клиентом (см. §4.4).

§3.8. Массовое добавление клиентов

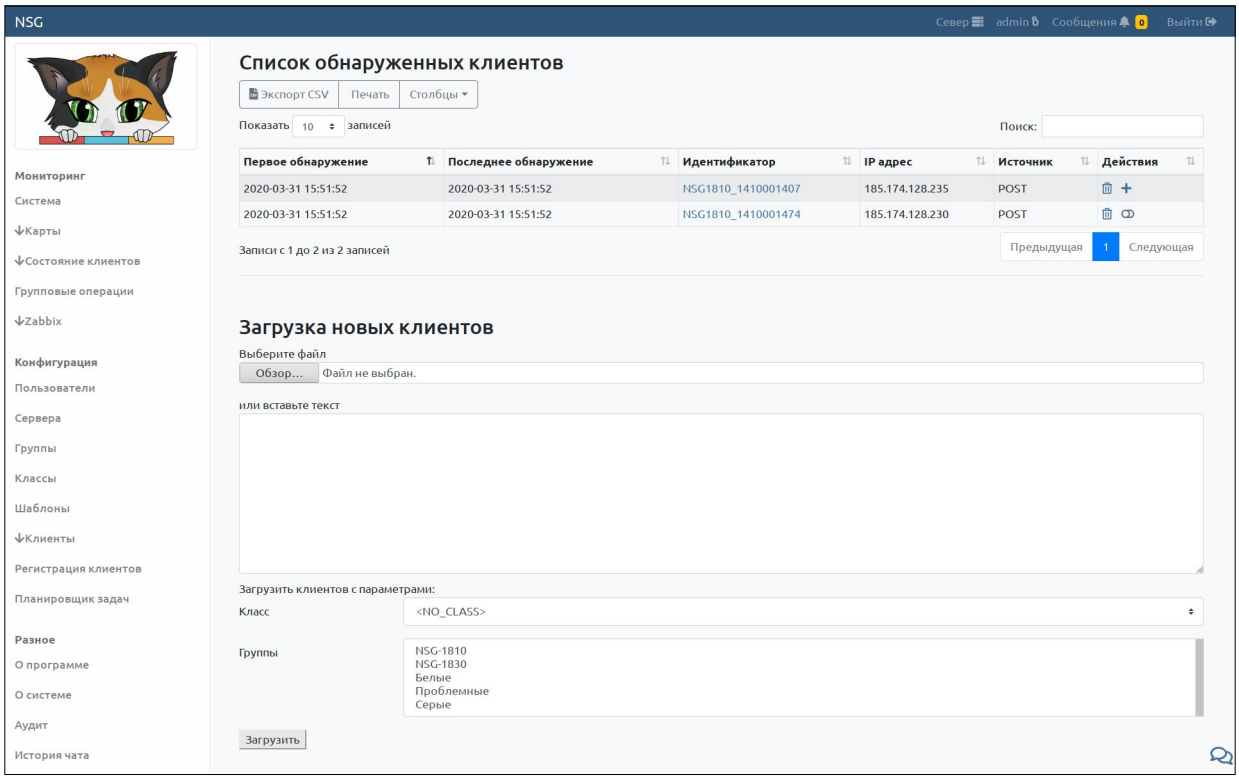
Страница "Регистрация клиентов" содержит несколько инструментов, которые позволяют упростить добавление новых клиентов в систему без ручного ввода длинных идентификаторов.

Если система получает запрос на конфигурацию от клиента, которого в ней не существует, то в таблице выводится его идентификатор и основная информация, известная о нём. Щёлкнув мышью на идентификаторе, можно просмотреть полную информацию. Если, например, администратор знает, что данный экземпляр оборудования действительно выдан сегодня технику, который поехал устанавливать его на площадке — он регистрирует этого клиента и далее настраивает его согласно §3.7. Если же он считает, что это какой-то подозрительный клиент, принадлежность и правомочность которого неизвестны — удаляет запрос из списка.

Если клиент был сконфигурирован заранее и заблокирован до ввода в эксплуатацию, то на месте кнопки "Добавить в систему" выводится кнопка "Разблокировать".

Партию устройств можно зарегистрировать в системе целиком, загрузив список устройств в виде одного файла. Файл может прилагаться в качестве сопроводительной документации или быть прислан по электронной почте.

Можно зарегистрировать в системе одно или несколько устройств, указав их идентификаторы в поле ввода. Это можно сделать не только с клавиатуры, но и любыми другими средствами — с помощью сканера QR-кодов, или *copy&paste* из файла или письма.



Формат для ручного ввода и для файла со списком клиентов:

ключ1=значение1, ключ2=значение2, ...

по 1 устройству в строке. В текущей версии "Богатка" поддерживаются ключи: id, dev, sn, причём обязателен либо первый из них, либо оба других, т.е. строки должны иметь вид

id=МОДЕЛЬ_ЗАВ.НОМЕР

либо

dev=МОДЕЛЬ, sn=ЗАВ.НОМЕР

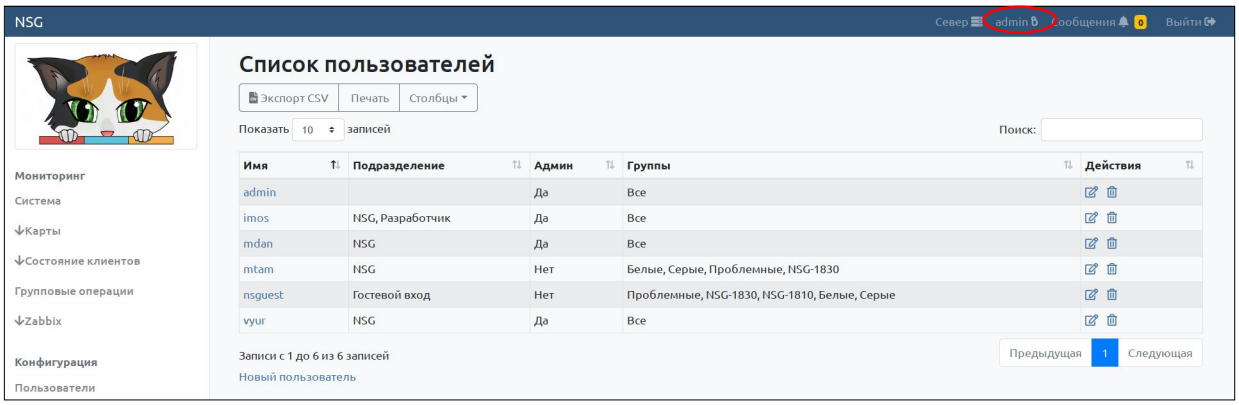
Также поддерживаются ключи latitude и longitude. В данном контексте, с их помощью можно установить местоположение по умолчанию для всех новых устройств — например, местный филиал организации-заказчика.

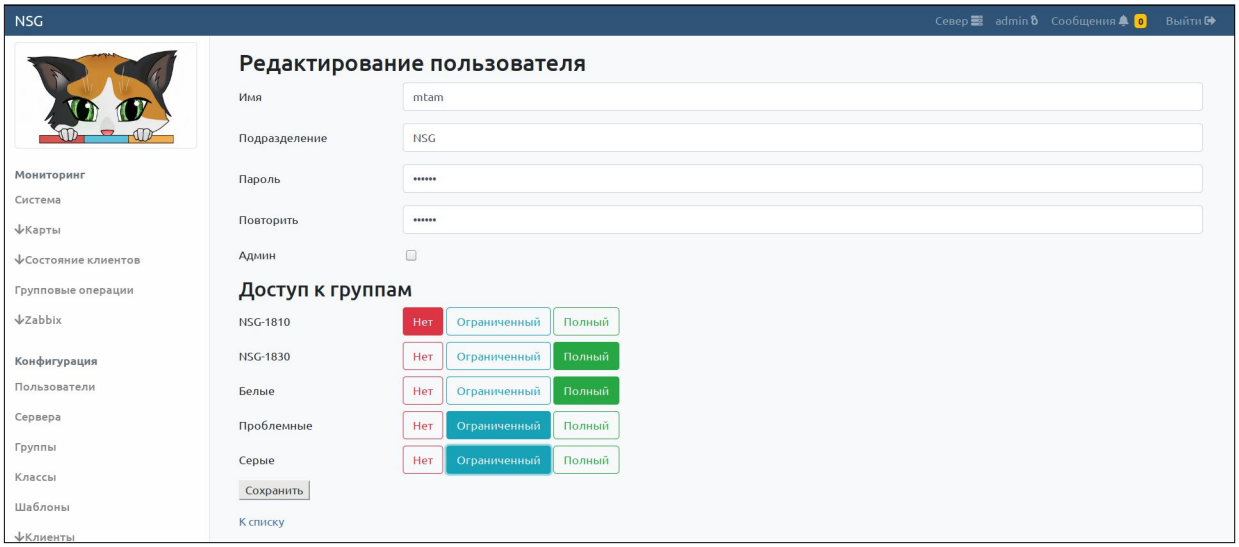
При загрузке файла или ручном вводе можно установить для всех новых клиентов класс и набор групп. Это удобно, например, когда известно, что данная партия оборудования состоит из клиентских устройств определённой модели и комплектации, будет подключаться по определённой схеме и администрироваться определёнными пользователями.

После регистрации одним из этих двух способов клиенты автоматически получают статус "Заблокирован". Разблокировать клиентов следует индивидуально, вручную, в момент их фактического ввода в эксплуатацию, одновременно с заполнением полей "Описание", "Адрес" и выполнением других сугубо индивидуальных настроек.

§3.9. Настройка пользователей

Пользователи в системе делятся на администраторов и юзеров обыкновенных (*userus vulgaris*). Администраторы имеют полные права доступа ко всем объектам в системе и ко всем действиям в ней. Пользователи не имеют доступа к операциям, которые касаются системы в целом. Им назначаются конкретные группы, к которым они имеют полный или ограниченный доступ, и они могут выполнять только управление и мониторинг в пределах своих групп.





Если клиент входит в несколько групп, по отношению к которым у пользователя есть разные права, то этому пользователю по отношению к этому клиенту даются максимальные из выданных прав.

Как частный случай, для демонстрационных и учебных целей можно создать пользователя guest с одними лишь ограниченными правами на одну или несколько групп.

Права пользователя на выполнение каких-либо действий в группах определяются с одной стороны, его правами на конкретного клиента, а с другой — правами, которые требует тот или иной конкретный скрипт (см. §3.4). Из этих двух ограничений берётся более жёсткое.

Доступные действия для различных категорий пользователей приведены в таблице.

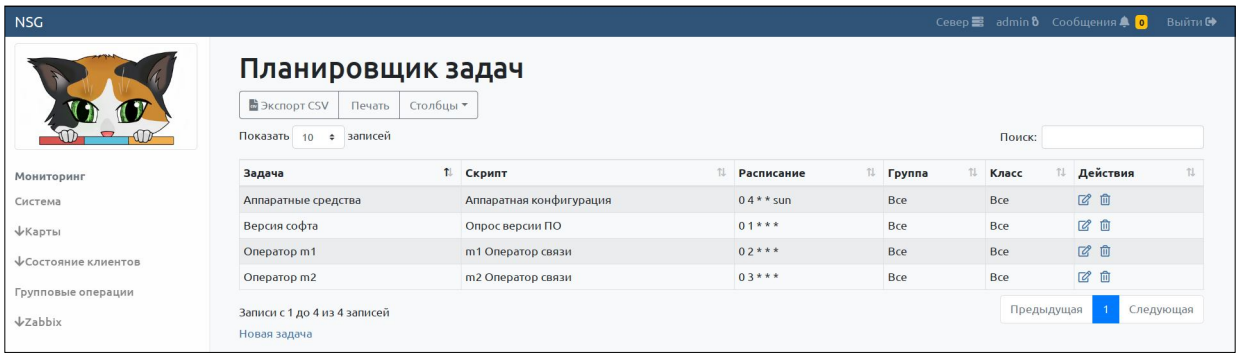
Действие		Админ.	Пользователь с правами на данного клиента:		
			Полными	Огранич.	Без прав
Удаление, редактирование клиентов		+	+		Не видит клиента
Блокировка, разблокировка, перезапуск клиентов		+	+	+	
Выполнение скриптов, требующих права:	админ.	+			
	полные	+	+		
	огранич.	+	+	+	

Вновь созданному пользователю необходимо назначить пароль. Поскольку это делает администратор, то при первом же входе в систему пользователь должен сменить пароль на уникальный сложный пароль, известный только ему. Страница редактирования пользователей обыкновенному пользователю недоступна; вместо этого он должен щёлкнуть мышью на своём имени, выведенном в заголовке страницы (отмечено на рисунках). Откроется ссылка на страницу изменения пароля, доступную только самому пользователю.

Если пароль пользователя утрачен или скомпрометирован, то администратор может самостоятельно изменить его; знать текущий пароль пользователя для этого не требуется. После этого пользователь должен снова сменить пароль на свой личный, как и в предыдущем случае.

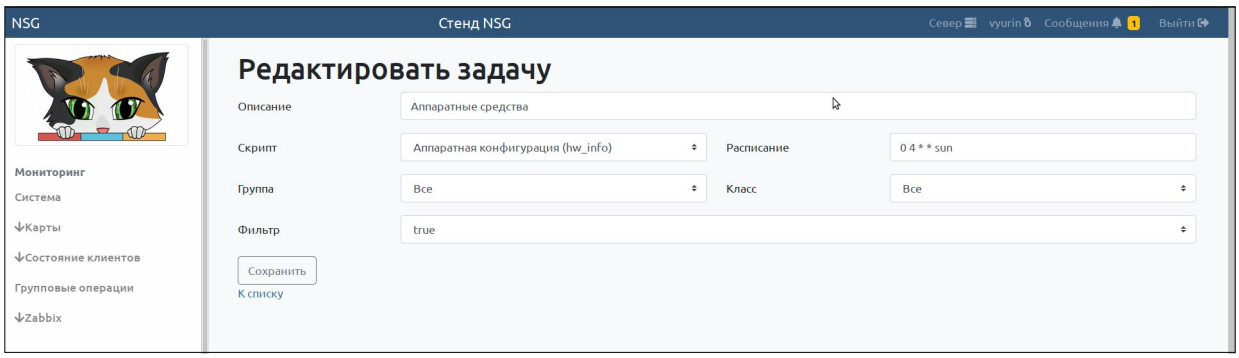
§3.10. Настройка планировщика задач

Планировщик позволяет выполнять скрипты в автоматическом режиме в заданное время, аналогично утилите *cron*. В первую очередь, это удобно для периодической инвентаризации парка клиентских устройств.



Предполагается, что каждый из скриптов, используемых в расписаниях, отсылает полученный ответ на сервер Zabbix (см. §3.4). Там эта информация накапливается и далее визуализируется, сортируется, фильтруется угодным пользователю образом.

Каждая задача содержит следующие поля:



Задача Название, оно же краткое текстовое описание задачи.

Скрипт Название исполняемого скрипта, из числа определённых для всех выбранных ниже классов (см. §3.4, §3.5).

Расписание Расписание исполнения данного скрипта. Записывается в формате *crontab* и состоит из 5 полей, разделённых пробелами: минуты, часы, дни, месяцы, дни недели. Каждое поле может содержать списки (через запятую), диапазоны (через дефис, только для числовых значений) и др. Подробно о формате см. *man 5 crontab*.

Группа Группа, по отношению к которой следует выполнить данный скрипт.

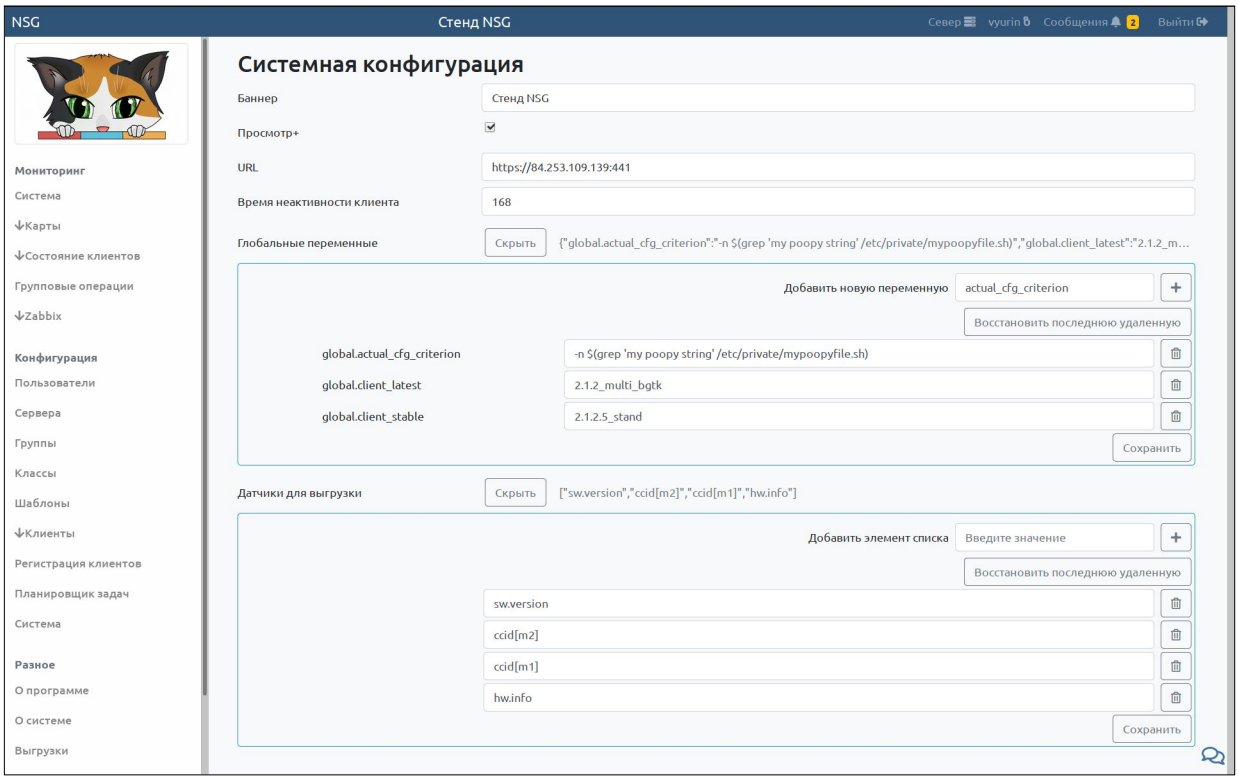
Класс Класс, по отношению к которому следует выполнить данный скрипт.

Фильтр — Фильтр, который можно использовать для дополнительного отбора клиентов из заданной группы и класса. Поскольку задача будет выполняться в автоматическом режиме и последствия ошибок могут быть весьма разрушительными, то прямое составление и редактирование фильтров в данном поле не предусмотрено. Допускается лишь выбирать из списка фильтров, которые составлены и отлажены на странице групповых операций (см. §4.6), а именно:

— личные фильтры данного пользователя
— фильтры, объявленные публичными
— тривиальный фильтр true (установлен по умолчанию, ни на что не влияет)
— существующий фильтр, установленный в данном поле ранее. Если этот фильтр будет впоследствии удалён из системы, то в настройках задачи всё равно сохраняется его копия (до следующего изменения данного поля).

§3.11. Общесистемные настройки

Страница "Системная конфигурация" (раздел меню Конфигурация —> Система) предусмотрен для разнородных общесистемных настроек, которые реализуются по мере развития системы или могут быть реализованы в будущем. Набор этих параметров и, как следствие, внешний вид страницы находятся в состоянии развития. Чтобы развернуть/свернуть описание параметра, следует щёлкнуть по нему мышью.



Баннер — Общий заголовок для всех страниц Web-интерфейса.

Просмотр+ — Разрешить пользователям с ограниченными правами редактировать справочные поля клиента (имя, адрес, географическое положение). Например, такими пользователями могут быть менеджеры в филиалах, непосредственно определяющие места установки клиентов.

URL — Базовый URL для внешних ссылок на страницы системы. В частности, необходим для доступа к инженерным страницам (см. §5.4).

Время неактивности клиента

Автоматический перезапуск клиента по истечении заданного времени. Данный способ следует использовать в решениях, где никаких процедур для ручного удаления клиентов административно не предусмотрено.

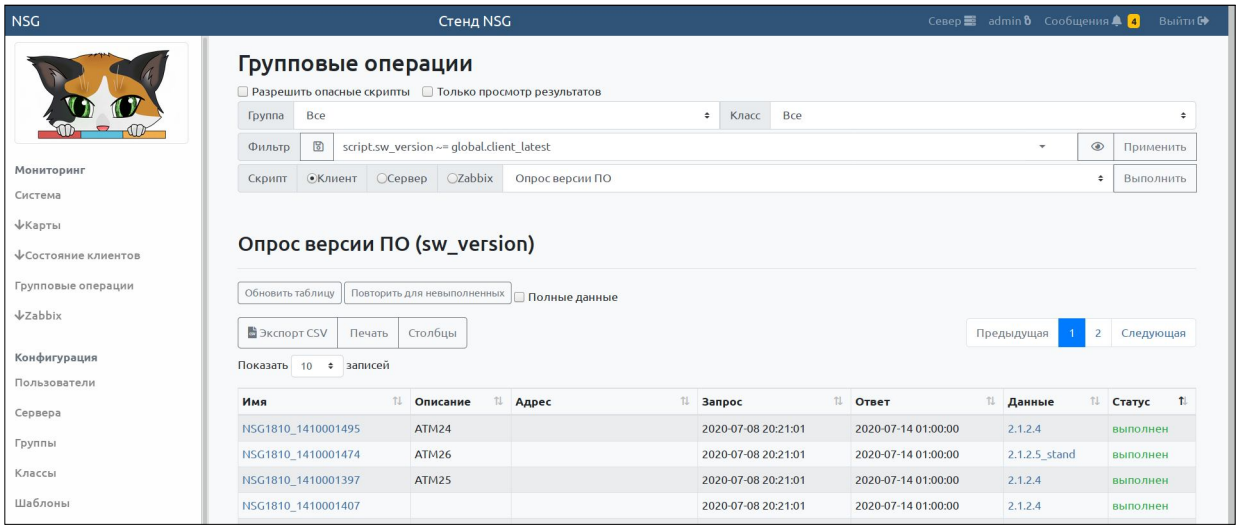
Предполагается, что если никаких мер за это время не принято — значит, так и должно быть. В противном случае, по мере вывода клиентов из эксплуатации они все будут накапливаться в состоянии "аварийные", и этот список будет расти неограниченно. Перезапуск возвращает клиентов в состояние "ни разу не подключавшиеся" и удаляет их аварийные сообщения.

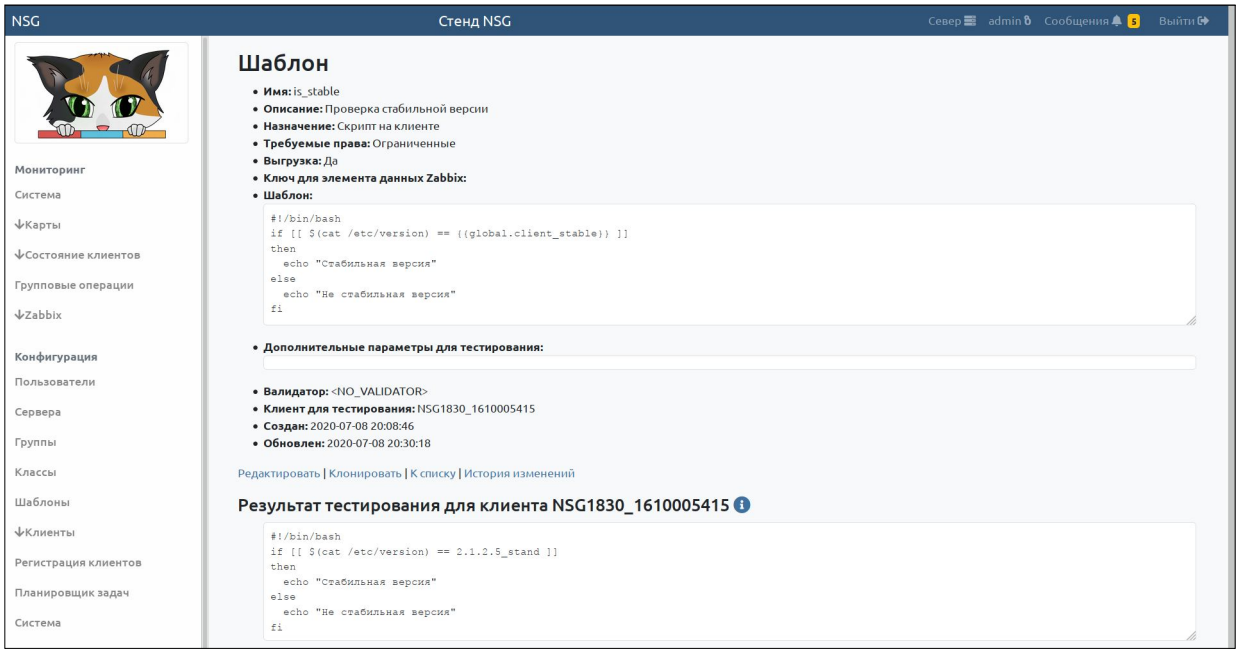
Время указывается в часах (напр. 168 часов = 1 неделя) и должно быть целым числом больше нуля. При любом другом значении (0, пустое поле, никогда и т.п.) автоматический перезапуск отключается.

Глобальные переменные

Переменные, которые единообразно относятся к системе в целом и не привязаны к конкретным классам устройств. Например, такими переменными могут быть номера стабильной и экспериментальной версии ПО. Тогда при смене текущей версии достаточно будет изменить номер только в системных настройках и не править его отдельно в каждом месте, где он используется.

Глобальные переменные можно использовать как в фильтрах, так и в шаблонах:





Датчики для выгрузки

Список датчиков, которые будут доступны на странице выгрузок (см. §5.5). Рекомендуется включать в этот список не все датчики, имеющиеся на клиенте, а только те, которые представляют интерес по существу. Как правило, это не сиюминутные, быстроменяющиеся показатели (уровень сотового сигнала, объём трафика и т.п. — это всё надо подсчитывать, подытоживать и усреднять средствами Zabbix), а относительно стабильные параметры, такие как инвентаризационные. Типичные примеры: версия ПО, версия материнской платы, тип, прошивка и IMEI сотовых модулей, ICCID, IMSI SIM-карт и названия сотовых операторов.

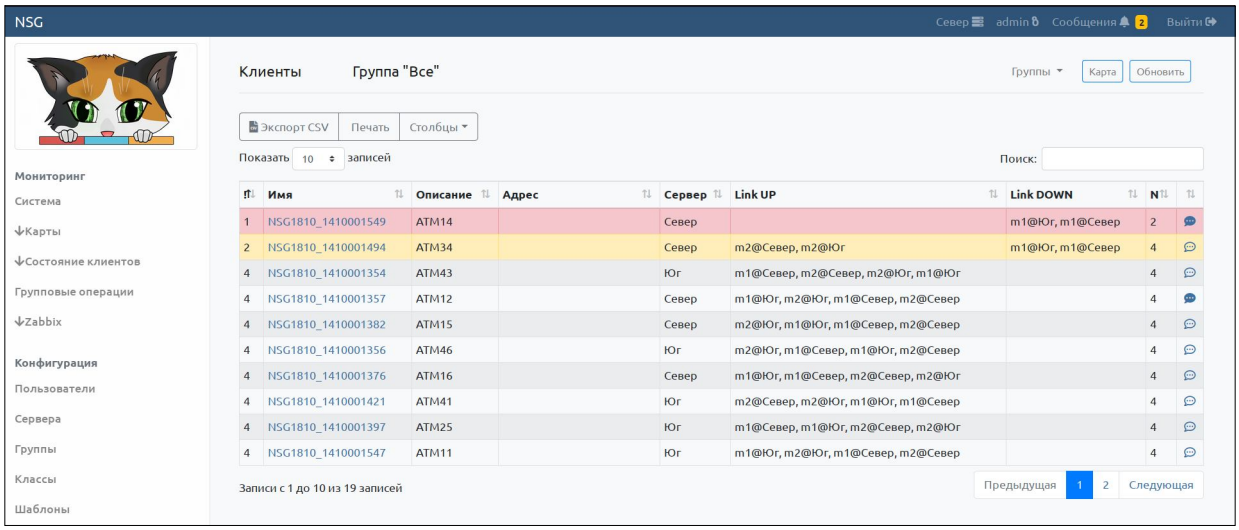
§4. Мониторинг и управление клиентами

§4.1. Состояние системы

На заглавной странице системы (в меню — пункт "Система") приводится сводная информация о системе. Содержимое страницы описано в §3.2.

§4.2. Мониторинг клиентов (таблица)

Подменю "Мониторинг —> Состояние клиентов" позволяет выбрать для работы часть клиентов, которым ранее была назначена определённая группа (из числа групп, доступных данному пользователю), либо "Все" для работы со всеми доступными клиентами. В основном поле интерфейса выводится таблица с состоянием всех клиентов, входящих в выбранную группу. Столбцы таблицы:



! Состояние клиента. Указывается цифрой и цветом строки. Для существенных состояний цвет соответствует индикации сообщений в правом поле (см. §3.2):

нет (тёмно-серый)	Клиент зарегистрирован в системе, но ни разу не выходил на связь.
0 (светло-серый)	Клиент зарегистрирован в системе, но ни разу не выходил на связь с момента последнего рестарта всех серверов (например, для обновления ПО).
1 (красный)	Клиент недоступен.
2 (жёлтый)	Клиент недоступен для одного из серверов по всем каналам связи, или для всех серверов по одному из каналов связи.
3 (синий)	Клиент недоступен для некоторых серверов по некоторым каналам связи.
4 (белый/серый)	Штатное рабочее состояние: клиент доступен для всех серверов по всем каналам связи.

Первоначальная сортировка при открытии страницы производится по этому полю, т.е. первыми выводятся клиенты с наиболее тяжёлыми аварийными состояниями.

Имя Идентификатор клиента.

Описание Описание клиента.

Адрес Адрес клиента.

Сервер Сервер, назначенный клиенту в качестве основного в данный момент.

Link UP
Link DOWN
: Список работающих и не работающих, соответственно, соединений — с каждым сервером по каждому из каналов связи.

N
: Число обнаруженных (работавших когда-либо) TCP-соединений с данным клиентом.

без имени
: Кнопки для доступа к комментариям и сообщениям искусственного интеллекта для каждого клиента (см. §3.7).

Специфические органы управления в данном окне:

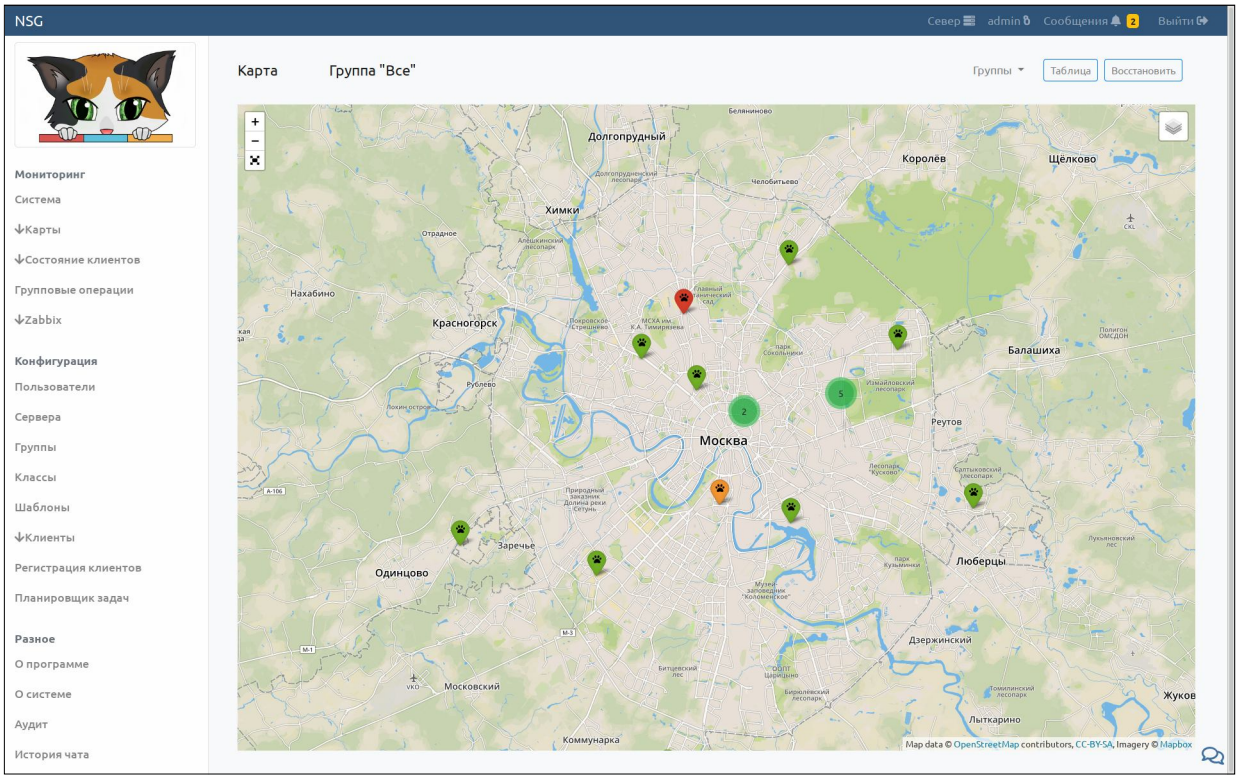
Кнопка Обновить
: Обновить содержимое таблицы. Автоматическое обновление на данной странице не производится, поскольку при изменении состояния соединений строки менялись бы местами, и с таблицей сложно было бы работать.

Кнопка Карта
: Переход на страницу мониторинга клиентов на карте (см. §4.3).

Выпадающий список Группы
: Выбор для просмотра одной заданной группы, либо всех клиентов, видимых пользователю. В списке содержатся все группы, к которым он имеет полный или ограниченный доступ.
Этот же результат можно получить, если сразу выбрать в меню в левой панели пункт "Клиенты по группам" и соответствующий подпункт в нём.

§4.3. Мониторинг клиентов (карта)

Подменю "Мониторинг —> Карты" позволяет выбрать для работы часть клиентов, которым ранее была назначена определённая группа (из числа групп, доступных данному пользователю), либо "Все" для работы со всеми доступными клиентами. В основном поле интерфейса выводится карта с указанием местоположения всех клиентов, входящих в выбранную группу.



Первоначальный масштаб карты выбирается таким образом, чтобы в поле изображения уместились все выбранные клиенты.

Одиночные клиенты изображаются указателем, цвет которого соответствует состоянию клиента. Если несколько клиентов не разрешаются в масштабе карты, то они изображаются кластером, который может дробиться на части или сливаться с другими указателями и кластерами по мере изменения масштаба. Цвет кластера устанавливается по худшему из состояний входящих в него клиентов.

При щелчке мышью по указателю выводится имя и описание клиента. При наведении мыши на кластер выводится многоугольник, указывающий расположение входящих в него клиентов. При щелчке мышью по кластеру масштаб карты изменяется таким образом, чтобы входящие в него клиенты разрешились в максимально возможной степени в имеющемся поле изображения.

Специфические органы управления в данном окне:

Кнопка Восстановить
: Восстановить первоначальный масштаб карты.

Кнопка Таблица
: Переход на страницу мониторинга клиентов в виде таблицы (см. §4.2).

Выпадающий список Группы
: Выбор для просмотра одной заданной группы, либо всех клиентов, видимых пользователю. В списке содержатся все группы, к которым он имеет полный или ограниченный доступ.

Кнопка Слои (в окне карты)
: Выбор поставщика картографической информации. Поддерживаются бесплатные сервисы общего пользования.

§4.4. Управление клиентом

При щелчке мышью по имени клиента в таблице или на карте открывается страница индивидуального мониторинга и управления им. Страница содержит следующие разделы:

Заголовок — информация о клиенте. Кнопка Zabbix позволяет перейти на страницу комплексных экранов данного клиента на сервере Zabbix. Кнопка Конфигурация клиента служит для перехода на страницу просмотра параметров клиента (см. §3.7) в разделе "Конфигурация".

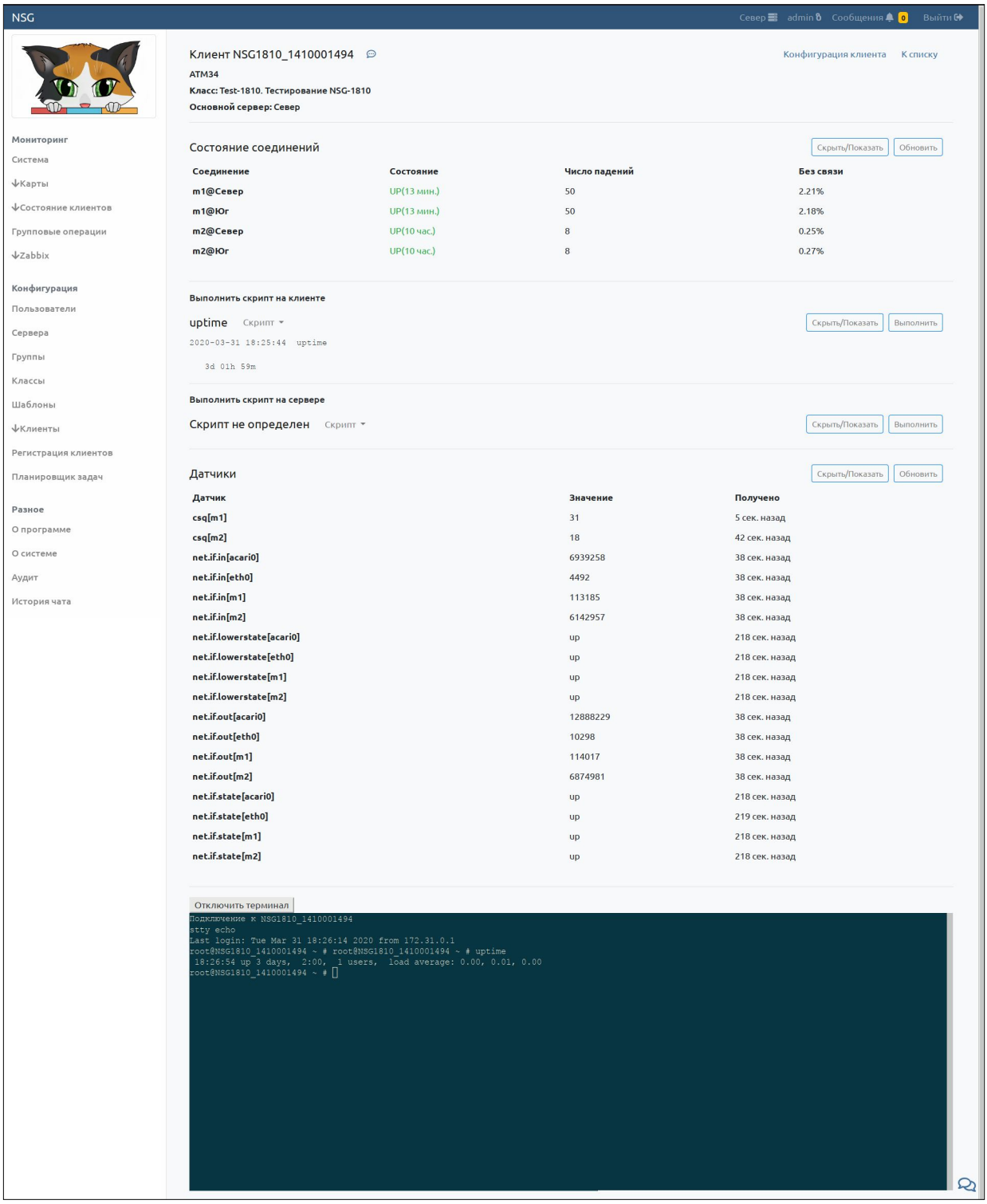
Состояние соединений — информация о состоянии соединений по каждому из каналов связи, имеющихся у клиента, с каждым из серверов системы. Кнопка "Скрыть/Показать" выводит весь раздел полностью или только его заголовок, для экономии места на экране. (Аналогично в последующих разделах.) Кнопка "Обновить" — обновляет содержимое раздела.

В столбце "Протокол" показывается выбранный протокол соединения: с шифрованием TLS v1.3 или открытый TCP (см. §7.1). При нажатии на ссылку выводится подробная информация о согласованных параметрах шифрования.

Выполнить скрипт на клиенте — вручную выбрать и выполнить скрипт. Выпадающий список содержит все скрипты, доступные одновременно для данного класса и для данного пользователя в соответствии с его правами на группы, в которые входит данный клиент. Если ранее выполнялся какой-либо скрипт, то он остаётся выбранным, а ниже выводятся время выполнения и его результат. Кнопка "Выполнить" — выполняет выбранный скрипт.

Выполнить скрипт на сервере — то же самое для скриптов, исполняемых на серверной стороне туннеля.

Выполнить скрипт Zabbix — то же самое для скриптов, которые исполняются на серверах Богатка и обрабатывают данные, хранящиеся в Zabbix.



ПРИМЕЧАНИЕ В системе всегда сохраняется результат только одного исполнения скрипта — при содержательном ответе, последнего. Если скрипт возвращает одни только пробельные символы или не возвращает ничего (например, по причине потери связи с клиентом), то сохраняется предыдущее значение.

Датчики — список программных датчиков, определённых на клиенте, и последних показаний, полученных от них. По существу, датчики являются Zabbix traps. При наступлении заданных событий (или истечении заданного интервала опроса какого-либо параметра, что также есть событие) они передают своё имя и состояние на сервер "Богатка", а он ретранслирует эти показания на сервер Zabbix. В самой системе "Богатка" в каждый момент времени хранится только одно последнее показание каждого датчика. Эта информация хранится только в оперативной памяти системы и не записывается на постоянные носители.

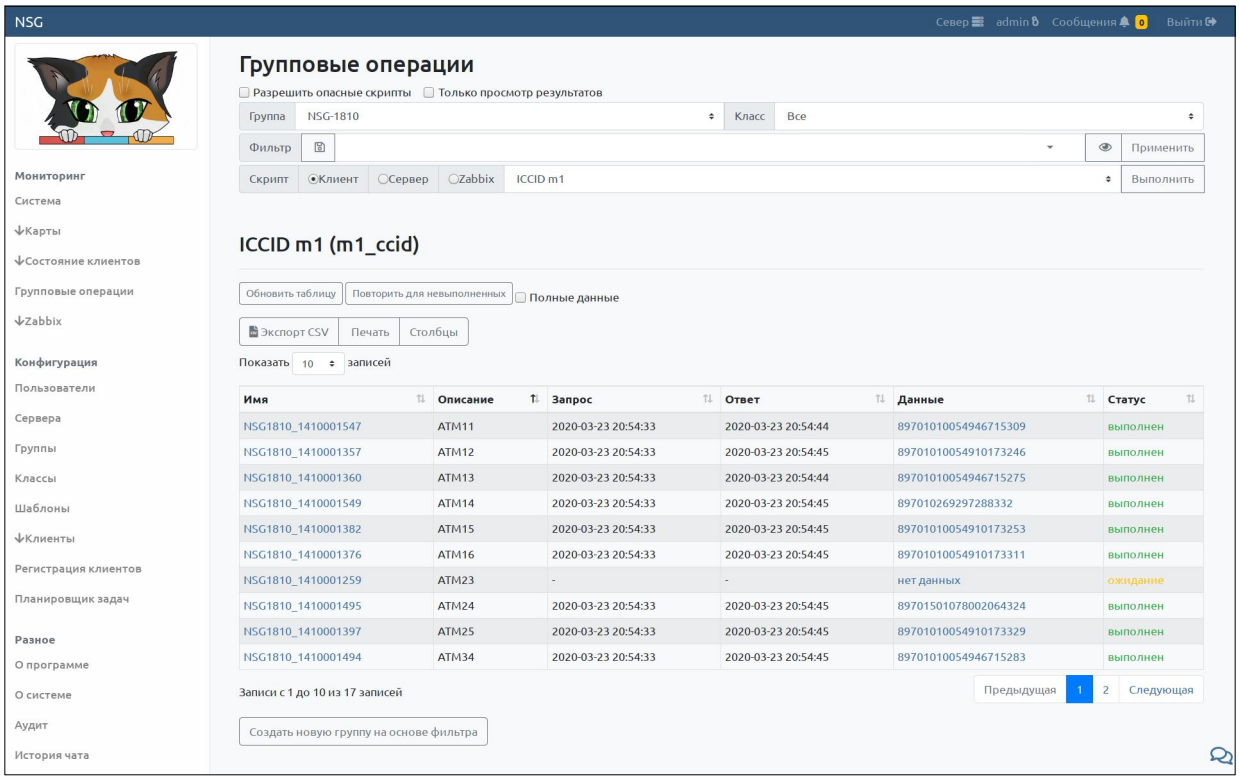
Подключиться к клиенту / Отключить терминал — кнопка открывает/закрывает терминальное окно и устанавливает SSH-соединение, по которому можно работать с клиентом в режиме его CLI непосредственно из Web-интерфейса системы. Для аутентификации при входе на клиента, в практических инсталляциях, в процессе конфигурации клиента ему передаётся ключ RSA сервера. Возможна и аутентификация с ручным вводом пароля при каждом входе.

ПРИМЕЧАНИЕ Вход на клиента осуществляется под именем root . На устройствах NSG Linux 2.*x* при этом рабочей командной оболочкой является *bash*. Фирменная оболочка *nsgsh* может быть вызвана, при необходимости, в пакетном или интерактивном режиме.

Дополнительные терминалы — определяются пользователем в конфигурации системы (см. §3.4) и позволяют выполнить заданную команду непоследственно после подключения по SSH. Например, такой командой может быть *nsgsh* — в этом случае пользователь попадает сразу в фирменную командную оболочку NSG. Или же это может быть вызов команды *nsgcu* на заданный асинхронный порт, с заданной скоростью и т.п. — в этом случае пользователь сразу попадёт прозрачно в указанный порт и сможет управлять оборудованием, которое к нему подключено. Следует напомнить, что во всех случаях пользователь находится на устройстве NSG в качестве root .

§4.5. Групповые операции

Страница "Групповые операции" позволяет выполнять скрипты мониторинга и управления одновременно для большого числа клиентов. Скрипты могут исполняться на клиенте, на сервере применительно к нему самому, или на сервере "Богатка" применительно к данным, полученным с сервера Zabbix.



С помощью данного механизма, например, можно:

— Опросить текущую версию ПО на всех клиентах (или в некотором классе и/или группе).
— Выбрать клиентов, у которых версия ПО ниже некоторой (например, в момент предыдущего обновления они были выключены).
— Повторно запустить для этих клиентов скрипт обновления ПО.

Первые три поля "Группа", "Класс" и "Фильтр" предназначены для выбора туннелей, для которых будет выполняться некоторый скрипт. Туннели, соответствующие всем 3 критериям одновременно, отображаются в таблице ниже. Фильтр позволяет отобрать для дальнейшей работы только тех клиентов, которые соответствуют некоторым условиям по результатам выполнения предыдущих скриптов. Подробно о фильтрах см. §4.6. При первом заходе на данную страницу указанные поля — пустые и никакие клиенты не отображаются.

В строке "Скрипт" необходимо:

- Выбрать объект, к которому будет применяться желаемый скрипт: клиентское устройство, либо серверная сторона туннеля, либо сервер Zabbix. В зависимости от этого выбора, предлагается список скриптов (из числа доступных для данного класса клиентов и для данного пользователя).
- Выбрать собственно скрипт.
- Нажать кнопку "Выполнить" для выполнения скрипта.

Результаты выполнения скрипта отображаются в таблице. Если скрипт исполнялся ранее, то в таблице изначально показываются данные от предыдущего исполнения. Чтобы увидеть результаты текущей попытки, необходимо нажать кнопку "Обновить таблицу". (Если скрипт исполняется долго или ответы приходят по медленным каналам связи, то можно нажимать её несколько раз и видеть, как таблица заполняется полученными ответами.) Если ответ от каких-либо клиентов не получен по истечении разумного времени (для данного скрипта), то скрипт можно избирательно выполнить ещё раз только для них с помощью кнопки "Повторить для невыполненных". После того, как от всех клиентов получен ответ на данную попытку, эти две кнопки скрываются.

В поле "Данные" таблицы выводится ответ клиента или, если ответ длинный, начальная его часть. Чтобы просмотреть ответ полностью, следует нажать мышью на это поле, или использовать следующую опцию.

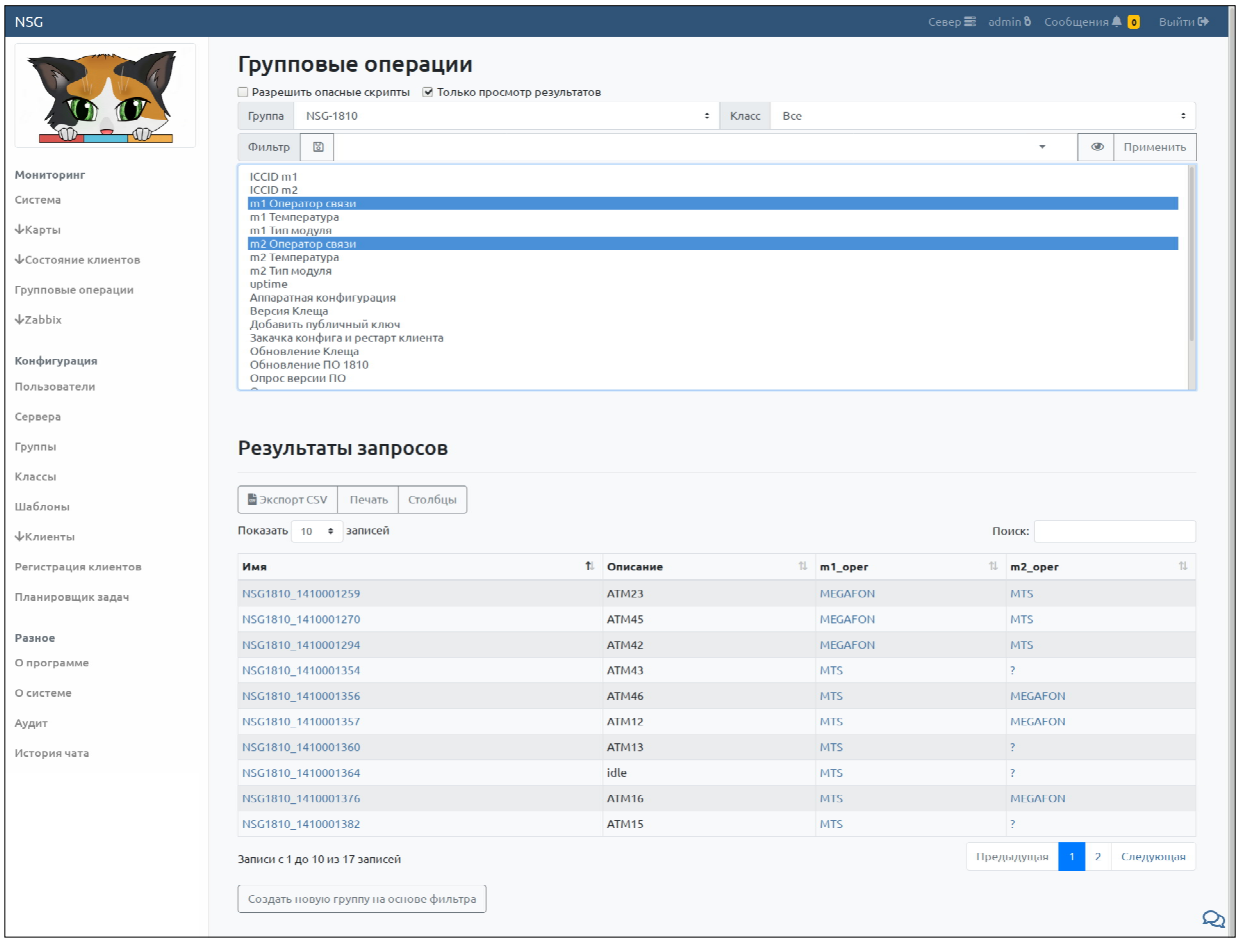
Опция "Полные данные" выводит ответы клиентов в таблицу полностью, невзирая на их длину и возможный неудобный вид таблицы. Опцию следует использовать, в основном, при экспорте таблицы на печать или в файл .csv, чтобы не потерять обрезанную информацию.

Опция "Разрешить опасные скрипты" показывает в списке скриптов те, имена которых начинаются с точки. Эти имена рекомендуется использовать для критически ответственных и потенциально опасных действий, таких как обновление ПО или конфигурации клиентов. По умолчанию опция выключена, чтобы такие скрипты нельзя было выполнить случайно. Это элемент безопасности и целостности системы, тем более, при массовом выполнении операций.

В режиме "Только просмотр" можно, как следует из его названия, только просмотреть ответы скриптов, выполненных ранее. Зато в этом режиме можно вывести результаты нескольких скриптов сразу. (Для выбора из списка используются Shift-click и Ctrl-click .) Например, если на клиенте установлены два сотовых интерфейса, то можно просмотреть результаты аналогичных скриптов сразу по обоим.

Кнопка "Создать новую группу на основе фильтра" позволяет администратору образовать новую группу из всех клиентов, видимых в таблице (на всех её страницах) в данный момент. Для группы автоматически генерируется имя вида _TMP_########### и описание в формате Группа \ Класс \ Фильтр; то и другое можно далее редактировать по своему усмотрению. Например, администратор может выбрать всех клиентов, у которых уровень сотового сигнала на одном или другом интерфейсе ниже некоторого порога, сформировать из них временную группу и передать права для работы с ней сотруднику, который будет решать эту проблему.

ПРИМЕЧАНИЕ С формальной точки зрения, временная группа, созданная указанным образом, ничем не отличается от групп, созданных вручную (см. §3.6).



§4.6. Фильтры для групповых операций

Фильтры используются для отбора более узкой группы клиентов по результатам выполнения скриптов (и/или с учётом их индивидуальных параметров). Фильтр представляет собой выражение на языке Lua 5.2, которое применяется к каждому клиенту из числа тех, которые входят в указанный класс и групп. Если значение выражения не есть nil или false , клиент оставляется в списке для дальнейшей работы, если одно из этих двух — исключается из рассмотрения.

Данные. Фильтры оперируют данными из 3 структур: client, script и global. Первая структура имеет следующие поля:

client.name — строка, имя клиента
client.description — строка, описание клиента
client.groups — строка, список групп (см. ниже)
client.lock — булевская переменная (true/false), блокировка клиента
client.latitude — число, широта местоположения
client.longitude — число, долгота местоположения
client.params — структура, индивидуальные параметры клиента
client.params.*ПАРАМЕТР* — строка, значение индивидуального параметра клиента (см. §3.5, §3.7)

Все значения структуры client берутся из текущей конфигурации клиента. Поле client.lock в данном случае является формальным, на самом деле оно учитывается всегда. Заблокированные клиенты в списке не показываются. Поле client.groups представляет собой список следующего вида:

```
client.groups = {
  [1] = true,
  [2] = true,
  [5] = true
}
```

где ключ — это идентификатор группы (указан на странице мониторинга групп и свойств группы, см. §3.6), а значение всегда true. Таким образом, если использовать в фильтре выражение client.groups[n], то оно возвращает true, если клиент принадлежит группе n, и возвращает nil, если не принадлежит. Комбинируя несколько полей, можно создать фильтр с любой логикой, например:

```
client.groups[3] or client.groups[2]
client.groups[3] and client.groups[5]
client.groups[3] and not client.groups[1]
```

Структура script имеет поля, имена которых совпадают с именами шаблонов скриптов, а значения равны результату последнего исполнения данного скрипта. Тип значения — строка, но если строка представляет собой запись числа, то тип автоматически преобразуется в число. Если имя элемента структуры не совпадает ни с одним именем шаблона скрипта, то значением будет nil . Если на скрипт не получено никакого ответа (например, клиент недоступен, в таблице пишется "нет данных"), то значением также будет nil .

Структура global содержит глобальные параметры, определённые на странице системных настроек (см. §3.11).

Если имя параметра или скрипта содержит только буквы, цифры и знак подчёркивания (_), то его можно писать без кавычек, через точку. Если оно содержит какие-либо другие спецсимволы, то его необходимо писать только в кавычках в квадратных скобках. Примеры:

```
client.params.mode
client.params["radio mode"]
script.m1_oper
script["ip_ro.sh"]
```

Константы. В основном, это строки, с которыми сравнивается ответ скрипта. Если строка содержит спецсимволы, то их необходимо вводить в виде *escape*-последовательностей, например, \" . Другой вариант сравнения строк — с помощью функции match (см. ниже).

Операторы. Для работы с данными доступны следующие основные операторы:

Арифметические:	+	сложение
	–	вычитание
	*	умножение
	/	деление
	%	остаток от деления
	^	возведение в степень
	– (унарный)	смена знака на противоположный
Сравнения:	==	равно (по типу и значению)
(в т.ч. побайтовое	~=	не равно
сравнение для	<	меньше
строк)	>	больше
	<=	меньше или равно
	>=	больше или равно
Логические:	and	логическое И
	or	логическое ИЛИ
	not	логическое НЕ

Подробнее об операторах Lua и особенностях их действия см.:

https://www.lua.org/manual/5.2/manual.html#3.4

ПРИМЕЧАНИЕ Операторы сравнения чисел выдают ошибку, если аргументы имеют разный тип — например, если скрипт вернул, вместо числа, пустую строку или "нет данных". Чтобы обойти её, следует делать дополнительную проверку с помощью функции type. Например (и именно в таком порядке):

```
type(script.m1_csq) == "number" and script.m1_csq < 10
```

Функции. В выражениях могут использоваться любые функции языка Lua 5.2, кроме модулей io, file и os . Подробнее см.:

https://www.lua.org/manual/5.2/contents.html#index

В частности, для поиска в строках может использоваться функция match(СТРОКА, ШАБЛОН) из модуля Wild. Шаблон может содержать обычные подстановочные символы и регулярные выражения, в частности:

*	любое число (в т.ч. 0) любых символов
?	ровно один любой символ
[abc]	любой символ из перечисленных
[a-z]	любой символ из диапазона
[!...]	любое, не совпадающее с указанными токенами

Примеры:

match("foobar", "foo*")	результат: true
match("foobar", "fo[a-z]bar")	результат: true
match("foobar", "bar*")	результат: false

Подробно о данной функции см. *https://hexdocs.pm/wild/Wild.html#match?/3* .

Другие употребительные функции: tostring, tonumber, type, модуль math.

Фирменные (проприетарные) функции. Специально для языка фильтров определена функция сравнения строк в формате, характерном для номера версии NSG Linux:

```
vercmp(verA, verB)
```

Функция сравнивает две строки в формате "X.Y.Z...", где X, Y, Z — числа. Количество чисел может быть любым, остальные символы в строке игнорируются. Функция возвращает одну из строк "lt" , "gt" или "eq" в зависимости от результата сравнения. Примеры (все приведённые выражения дают значение true):

```
vercmp("2.1.9", "2.1.10") == "lt"
vercmp("2.1.2", "2.1.1.1") == "gt"
vercmp("2.1.2", "2.1.2_pre1") == "eq"
```

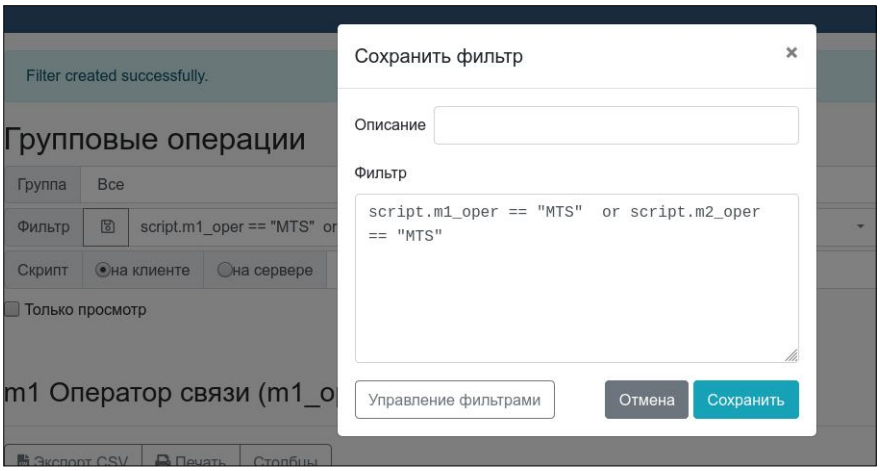
Примеры фильтров:

```
script.m1_oper == "MTS" or script.m2_oper == "MTS"
client.latitude > 55.75
match(script.sw_version, "2.1.2*") and client.params.mode == "lte"
```

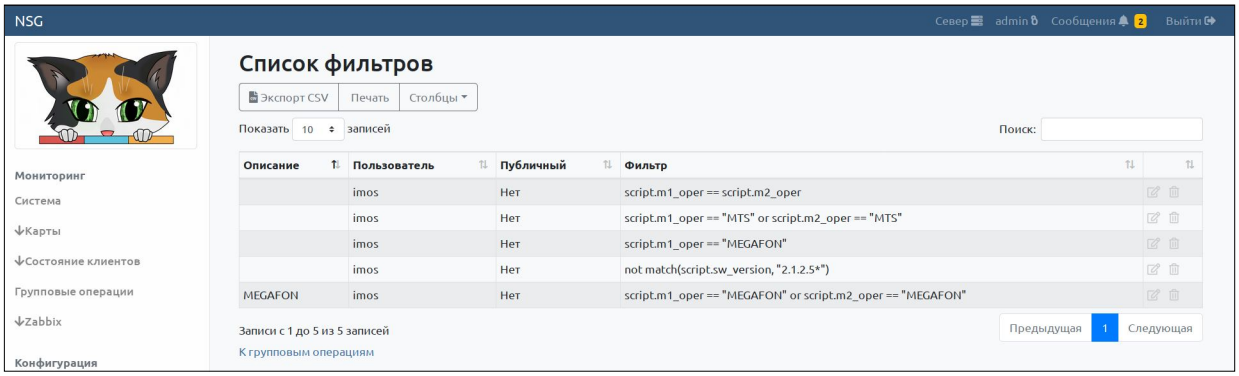
Если в строке фильтра есть ошибка, то она будет выводиться при нажатии кнопки "Посмотреть" или "Применить".

Кнопка "Посмотреть" предназначена для отладки строки фильтра. При ее нажатии выводится таблица с вычисленными значениями фильтра для каждого клиента. Например, если набрать в строке фильтра script.m1_oper , то кнопка покажет реальные тип и значения этой переменной. Если набрать в строке фильтра имя структуры, то будут выведены все поля структуры и их значения.

Созданный фильтр можно сохранить для последующего использования при помощи кнопки 🖫. При нажатии кнопки открывается окно для сохранения фильтра, в котором можно добавить описание (опционально). Эти фильтры можно в дальнейшем выбирать из выпадающего списка.



Набор фильтров хранится отдельно для каждого пользователя. В списке пользователь видит только свои фильтры и фильтры, объявленные администраторами как "публичные". Фильтры других пользователей он не видит.



Кнопка "Управление фильтрами" открывает страницу фильтров, где можно редактировать и удалять фильтры. Любой пользователь (в том числе администратор) может редактировать и удалять только свои фильтры. Но администратор:

— Видит все фильтры. Обычный пользователь видит только свои.
— Может присвоить статус "публичный" любому своему фильтру.

Расширенные возможности. Пользователи, знакомые с языком Lua, могут использовать и другие его возможности. При этом следует иметь в виду, что реально во встроенной Lua-машине исполняется скрипт

'return (<строка из поля Фильтр>)'

поэтому в фильтре может быть только выражение (*Lua expression*), значение которого и получается на выходе.

Если требуется написать в качестве фильтра более сложную программу, то следует использовать анонимную функцию, например:

```
(function( ) lon=client.longitude; if lon>37 and lon<38 then return true else return false end end)()
```

§4.7. Мониторинг в системе Zabbix

Сервер Zabbix функционирует отдельно от системы "Богатка" и аккумулирует всю телеметрию и инвентаризационную информацию, генерируемую клиентами или собираемую при помощи скриптов. Меню Zabbix имеет два подпункта:

Перейти
: Открыть Web-интерфейс сервера Zabbix в отдельном окне.

Синхронизировать
: Синхронизировать список клиентов и их основные атрибуты (имя, описание, принадлежность к группам). При штатной работе все изменения в конфигурации клиентов передаются в Zabbix автоматически и ручная синхронизация не требуется. Она предназначена только для случаев, когда были некорректно сделаны изменения вручную на самом сервере Zabbix или был сбой сети во время автоматической синхронизации. Синхронизация может выполняться в двух вариантах: быстрая и полная. Подробнее см. встроенную справку при выборе данного пункта меню.

При создании/удалении и блокировке/разблокировке клиентов их состояние в Zabbix изменяется по следующим правилам:

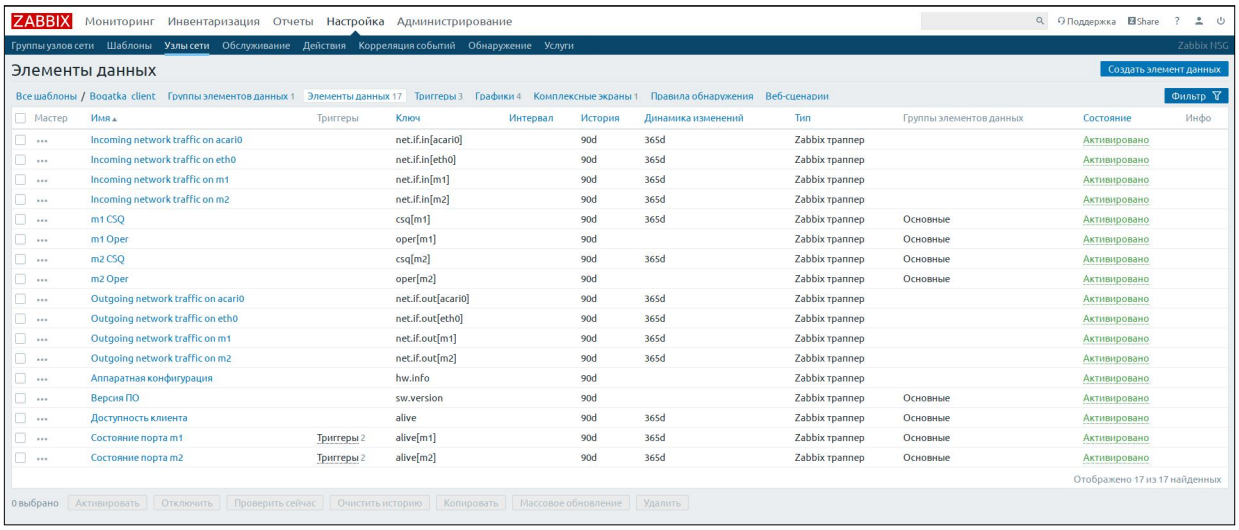
- При создании нового клиента (РАЗблокированного) — добавляется в Zabbix
- При создании нового клиента (ЗАблокированного) — НЕ добавляется в Zabbix
- При РАЗблокировании клиента — добавляется в Zabbix (если его там не было)
- При ЗАблокировании клиента — остается в Zabbix
- При удалении клиента — удаляется из Zabbix

При синхронизации состояние заблокированных клиентов в Zabbix не меняется.

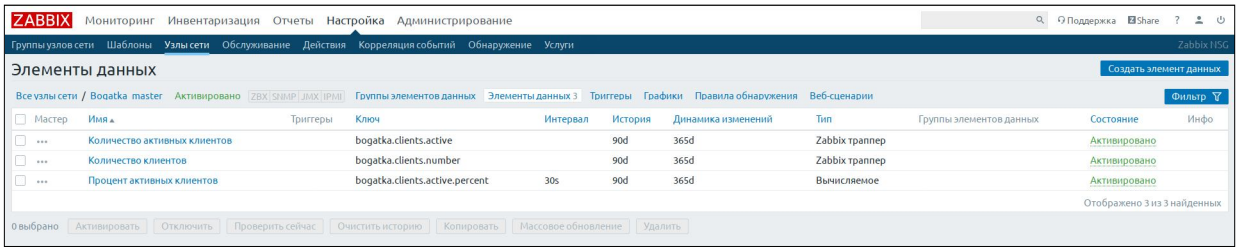
В типовом шаблоне Bogatka_client определён следующий набор метрик:

Название	Значения	Смысл
Доступность клиента (alive)	0/1	На связи клиент или нет
Состояние порта m1/m2 (alive[...])	0/1	Установлено ли соединение через данный порт
Версия клиента (sw_version)	строка	Номер версии ПО клиента
Аппаратная конфигурация (hw_info)	строка	Состав портов на данном клиенте
m1/m2 Oper (oper[...])	строка	Оператор сотовой связи для данного порта
m1/m2 CSQ (csq[...])		Уровень сигнала для данного порта
Incoming/Outgoing network traffic on *интерфейс* (net.if.in[...], net.if.out[...])	число	Входящий и исходящий трафик для каждого интерфейса (бит/с)

Состав метрик по необходимости может быть дополнен любым количеством элементов данных, которые генерируются в Богатке. Это статистика, результаты выполнения скриптов, значения всех датчиков на клиенте.

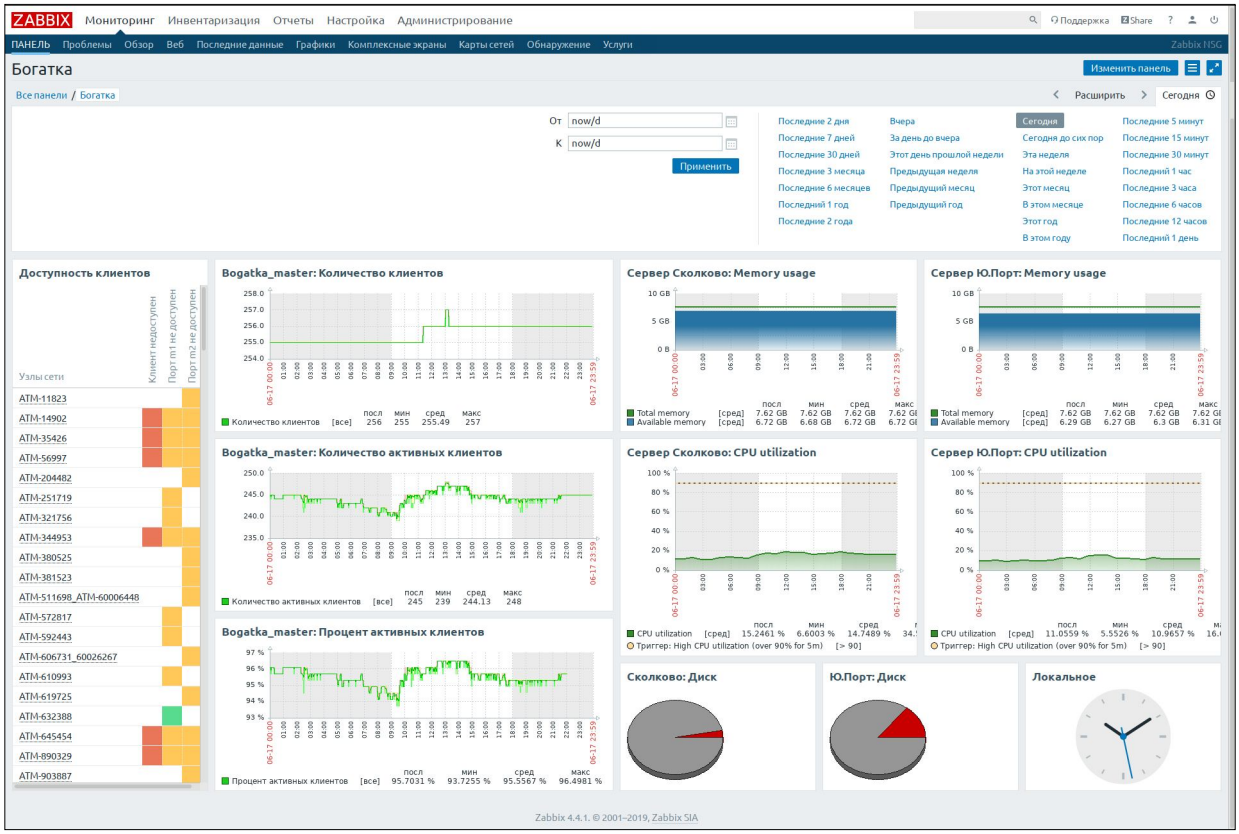


Кроме того, введен специальный узел сети Bogatka_master, который отражает общие характеристики системы. Для него определены 3 элемента данных, с очевидными названиями:

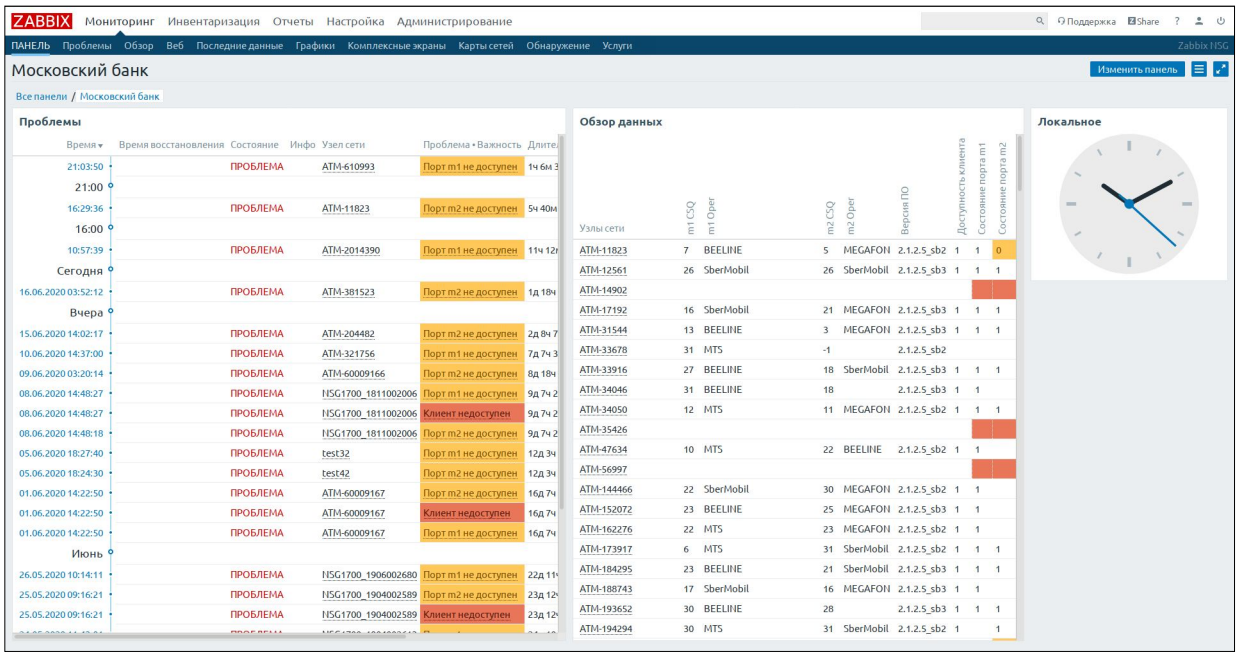


Для удобства просмотра в шаблоне определен комплексный экран. Например:

Также могут быть определены панели. К сожалению, в Zabbix невозможно экспортировать настройки панели. Их необходимо настраивать вручную в каждой инсталляции. Пример панелей для Богатки:



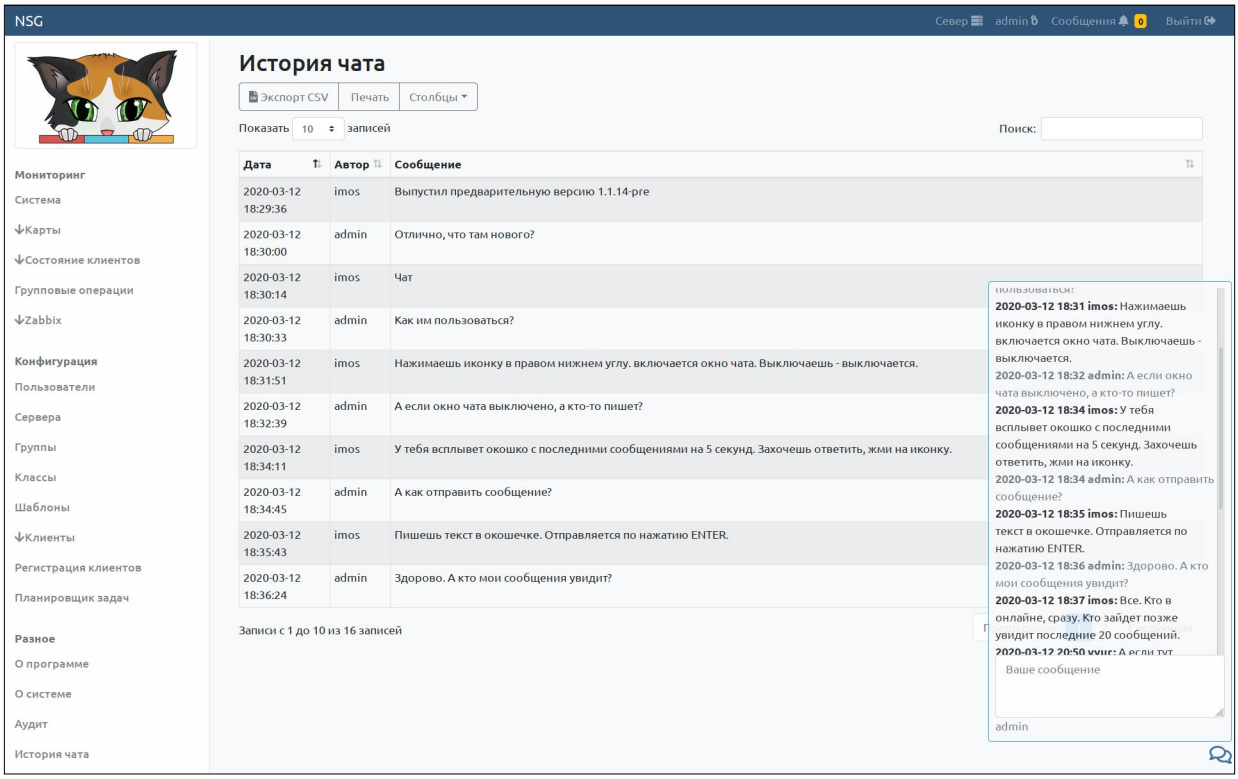
Для группы:



§5. Вспомогательные средства

§5.1. Пользовательский чат

Чат (по-русски — беседа) предназначен для быстрого обмена информацией между пользователями. Кнопка и окно чата расположены в правом нижнем углу и показываются поверх всех остальных элементов интерфейса. В окне чата показываются 20 последних сообщений, поле ввода и список пользователей, присутствующих в системе (т.е. номинально не вышедших из неё).



При входе в систему, если в чате есть новые сообщения (с момента предыдущего выхода данного пользователя), окно чата разворачивается на небольшое время, затем сворачивается.

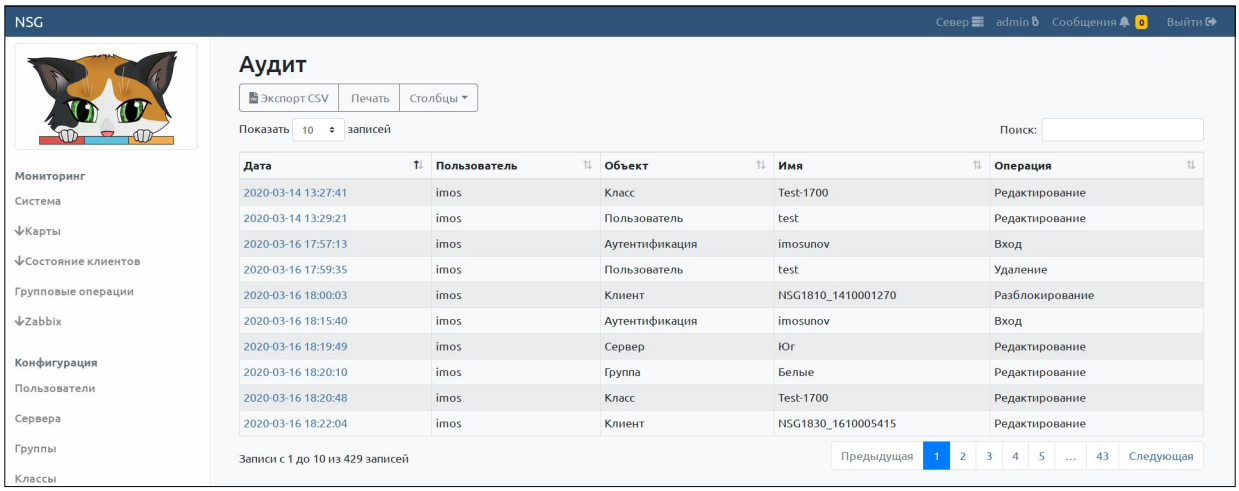
Чат предполагает обмен короткими однострочными сообщениями. Сообщение отправляется по нажатию кнопки Enter. Последующее редактирование и удаление сообщений не предусмотрено.

Сообщения за длительный период времени можно просмотреть на странице "Разное —> История чата". Как и на других страницах, здесь работают поиск, сортировка и др. сервисы.

ПРИМЕЧАНИЕ Чат не приватный, он доступен всем пользователям. Не всегда стоит говорить, что думаешь — но всегда надо думать, что говоришь.

§5.2. Аудит системы

Аудит системы является важным инструментом как для обеспечения безопасности, так и для своевременного поиска, устранения и дальнейшего предотвращения человеческих ошибок. Страница аудита доступна в разделе меню "Разное —> Аудит". На сводной странице выводится таблица: когда, кто, с чем и какое действие выполнил.

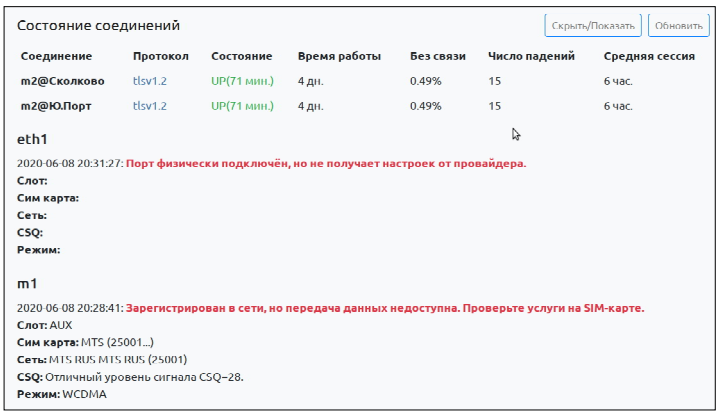


При нажатии на ссылку (в столбце "Дата") выводится подробная информация о данной записи, включая состояние объекта после выполнения данного действия.

§5.3. Искусственный интеллект

Искусственный интеллект, как и естественный, является полезным вспомогательным инструментом для выполнения разнообразных работ. Интеллект Богатки умеет самостоятельно диагностировать основные типы неполадок, исправлять человеческие ошибки, а также руководить пользователями, способными выполнять несложные механические действия, например: "заменить SIM-карту".

Сообщения ИИ выводятся на странице мониторинга клиента и на инженерной странице.



На сводной странице мониторинга для клиентов, у которых имеются сообщения ИИ, ставится дополнительная отметка — красный восклицательный знак в последней колонке. Чтобы прочитать сообщения, не уходя с этой страницы, следует навести курсор мыши на него. Аналогичным образом читаются пользовательские комментарии.

Технически ИИ Богатки реализуется в виде скриптов, исполняемых на клиентах. Сообщения ИИ отсылаются на сервер Zabbix в формате сообщений (*traps*) Zabbix с префиксом bogatka.wizard . Поскольку все сообщения идут не напрямую, а ретранслируются сервером Богатка, то он отфильтровывает сообщения с данным префиксом и обрабатывает их самостоятельно. В данной реализации Богатки предусмотрены следующие ключи:

bogatka.wizard.slot[ИНТЕРФЕЙС]
bogatka.wizard.sim[ИНТЕРФЕЙС]
bogatka.wizard.network[ИНТЕРФЕЙС]
bogatka.wizard.errormsg[ИНТЕРФЕЙС]
bogatka.wizard.csq[ИНТЕРФЕЙС]
bogatka.wizard.rat[ИНТЕРФЕЙС]

Все ключи представляют собой текстовые переменные, значения которых и определяются скриптом. Если состояние интерфейса нормальное, то ключ bogatka.wizard.errormsg должен быть пустым. Если этот ключ не пустой, то выводится текст аварийного сообщения и остальные параметры (даже если они пустые или не имеют смысла, например, для интерфейса Ethernet). Ключ bogatka.wizard.Action может содержать рекомендуемое действие и выводится в строку после bogatka.wizard.errormsg (см. снимок экрана выше).

Для отправки сообщений из скрипта используется утилита zabbix_sender . При составлении значений ключей необходимо двойное экранирование спецсимволов,

Пример скрипта: анализ состояния порта Ethernet. Скрипт вызывается с единственным параметром — именем порта.

```
ZBXZ=\"{{tun_pref}}.0.1\";              #адрес серверного конца туннеля
ZBXP=\"50051\";

if [[ $(ifconfig {%port%} | grep inet\\ addr:) ]]; then
  MSG=\"Порт подключён и настроен. Проблема выше у провайдера.\"
elif [[ $(ifconfig {%port%} | grep RUNNING) || $(ifconfig {%port%} | grep LOWER_UP) ]]; then
  MSG=\"Порт физически подключён, но не получает настроек от провайдера.\"
else
  MSG=\"Порт не подключён.\"
fi;
zabbix_sender -z $ZBXZ -p $ZBXP -s $HOSTNAME -k bogatka.wizard.errormsg[{%port%}] -o \"$MSG\" >/dev/null;
```

Дополнительно можно, например, вывести режим работы физического уровня, например:

```
RAT=$(ethtool {%port%} | grep -A1 Speed)
```

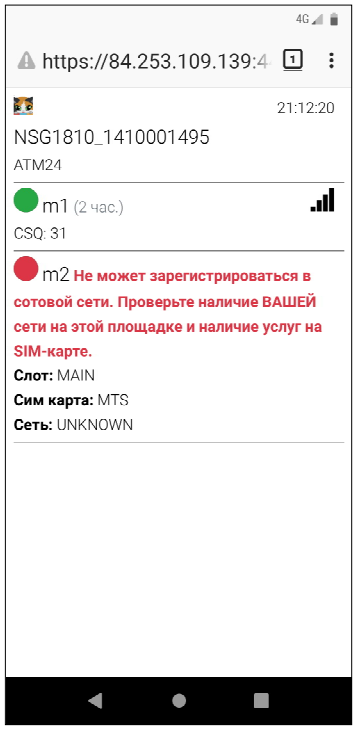
и записать его в ключ bogatka.wizard.rat[{%port%}] (он выведется в строке "Режим").

Чтобы отправить в одном сообщении несколько ключей, удобнее заранее упаковать их в одну переменную, например:

```
STATUS=\"- bogatka.wizard.slot[$1] \\\"$SLOT\\\"\\n\\
   - bogatka.wizard.sim[$1] \\\"$SIM ($MCC_MNC...)\\\"\\n\\
   - bogatka.wizard.network[$1] \\\"$MSG_OPER\\\"\\n\\
   - bogatka.wizard.errormsg[$1] \\\"$MSG\\\"\\n\\
   - bogatka.wizard.csq[$1] \\\"$MSG_CSQ\\\"\\n\\
   - bogatka.wizard.rat[$1] \\\"$RAT\\\"\\n\"
echo -en \"$STATUS\" | zabbix_sender -z $ZBXZ -p $ZBXP -s $HOSTNAME  -i-
```

Предполагается, что все приведённые переменные определены в скрипте ранее. Обратите внимание на двойное экранирование спецсимволов.

§5.4. Инженерная страница и печать QR-кодов



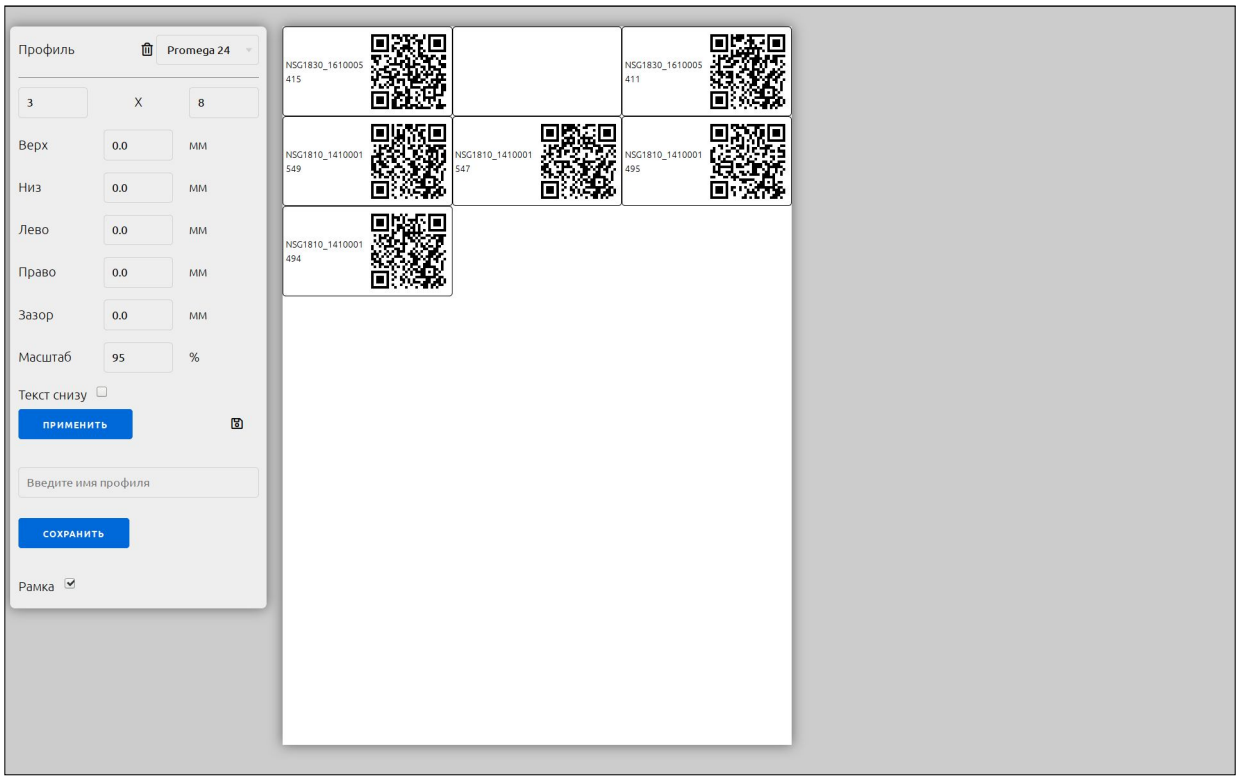
Инженерная страница содержит диагностическую информацию о клиенте, в первую очередь, о работе его каналов связи. Страница предназначена для инженеров и техников, подключающих оборудование на площадках, и позволяет им видеть состояние клиента самостоятельно, без обращения к оператору в процессе работы. Таким образом, техник может исправить ситуацию немедленно и покинуть площадку только после того, как убедится, что все каналы связи работают штатно. Например, если выяснится, что выбранная сотовая сеть недоступна в данном месте, то он может немедленно заменить карту на другого оператора. Таким образом, он избегает необходимости повторного выезда на площадку после того, как оператор обнаружит неисправности и сообщит ему об этом.

Инженерная страница содержит только ограниченный объём информации, необходимый технику для контроля, и не содержит никаких элементов, с помощью которых можно было бы получить не положенную ему информацию, или как-либо вмешаться в работу системы. Таким образом, её могут использовать не только зарегистрированные пользователи Богатки, но и сотрудники низового звена и даже сторонних организаций-подрядчиков.

Информация в броузере обновляется динамически по мере того, как она обновляется на сервере. Таймер в правом верхнем углу одновременно служит индикатором того, что соединение с сервером поддерживается.

Ссылка на инженерную страницу может быть передана исполнителю в виде QR-кода, наклеенного на устройство (см. §3.7), по SMS или иным способом.

Печать QR-кодов. При нажатии кнопки QR на странице конфигурации или регистрации клиентов открывается окно печати, в котором можно точно настроить макет страницы для печати на самоклеящихся этикетках конкретной модели, производителя, на конкретном принтере.



Упаковки листов с самоклеящимися этикетками продаются в магазинах канцелярских принадлежностей. Рекомендуется использовать этикетки размером 25–50 мм по короткой стороне.

ВНИМАНИЕ Поскольку большинство офисных принтеров технически не позволяет печатать без полей, рекомендуется использовать такие марки, где непосредственно этикетки занимают менее чем полный лист А4.

Рекомендуется

Не рекомендуется

Панель управления печатью (слева, на печать не выводится) позволяет настроить число рядов и столбцов на листе, поля со всех четырёх сторон, зазор между соседними этикетками. При этом собственно QR-код автоматически масштабируется до оставшегося доступного размера. При необходимости размер можно откорректировать вручную (в %% от расчётного размера), чтобы он гарантированно не выходил за границы этикеток с учётом неточностей подачи бумаги на конкретном принтере и т.п. Рекомендуется сначала распечать пробу на простом листе, сложить с этикетками и посмотреть результат на просвет.

Подпись можно разместить под QR-кодом, тогда то и другое будет расположено по центру этикетки. Но в этом случае размер QR-кода объективно вычисляется не очень точно (например, идентификатор клиента может занять 2 строки кроме одной, а её физическая высота может зависеть от настроек броузера), поэтому в большинстве случаев требуется подобрать его вручную.

Поле "Текст" позволяет разместить на каждой этикетке произвольный текст, например, инструкцию для инженера:

```
1. Отсканируй код.<br>2. Перейди по ссылке.<br>3. Убедись, что работают все порты.
```

По умолчанию, текст идёт сплошной строкой и переносится в любом требуемом месте. Для более специфичного оформления текста можно использовать HTML-теги. Пользователь должен самостоятельно заботиться о том, чтобы текст помещался в свободное поле этикетки и нормально читался.

Опция "Рамка" позволяет выводить на печать рамки отдельных этикеток или, наоборот, скрывать их.

Кнопка 🖫 позволяет сохранить подобранные настройки в виде готового профиля, который можно будет использовать для печати на этикетках этой марки, на этом принтере в будущем. Сохранённые ранее профили доступны в выпадающем списке вверху панели и видимы для всех пользователей.

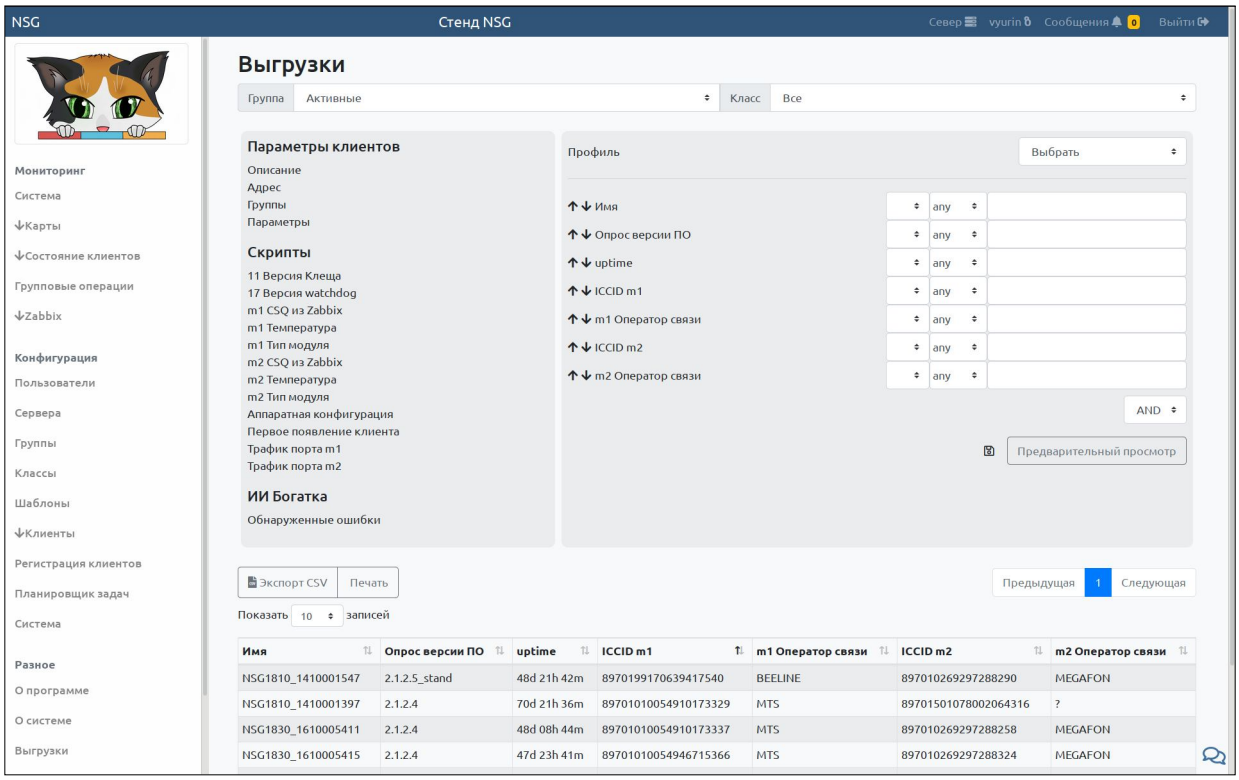
При нажатии мышью на любую из позиций в макете — листы перегенерируются, исключая/включая эту позицию. Это позволяет полностью использовать все этикетки на частично использованных листах, независимо от того, сколько этикеток и в каких именно позициях было израсходовано ранее.

Непосредственно для печати следует использовать функцию "Печать" Web-броузера.

ПРИМЕЧАНИЕ Генерация макета с большим количеством этикеток может занять заметное время. Это объективный факт, и это не является ошибкой.

§5.5. Выгрузки данных

Страница выгрузок позволяет экспортировать произвольные данные, доступные в системе, для всех клиентов или заданных групп и/или классов, в произвольном наборе и последовательности, с применением дополнительных фильтров.



Основное поле страницы содержит 2 панели. В левой показаны все данные, доступные для выгрузки (в зависимости от версии Богатки и от набора скриптов в данной инсталляции). При щелчке мышью по любому из полей оно перемещается на правую панель — в данные, подлежащие выгрузке. При щелчке мышью в правой панели указанные поля исключаются из выгрузки и перемещаются обратно на левую панель. Стрелки ↑ и ↓ позволяют изменять порядок следования данных, т.е. столбцов в таблице и полей в файле .csv.

Основные категории данных, доступные для выгрузки — это инвентаризационные параметры клиента, результаты выполнения скриптов и выводы, сделанные искусственным интеллектом. Телеметрия, полученная от датчиков, напрямую в выгрузку не включается, поскольку в Богатке она не хранится, только пересылается в Zabbix, а разовые последние показания дачиков, как правило, не представляют большого интереса. Для вывода телеметрии необходимо использовать скрипты для работы с Zabbix, которые запросят с сервера необходимую информацию и обработают её требуемым образом — например, вычислят среднее/минимальное/максимальное значение уровня сигнала (CSQ) за последние сутки.

ПРИМЕЧАНИЕ Страница выгрузок не выполняет никаких скриптов, а только экспортирует результаты скриптов, выполненных ранее. Чтобы получить с её помощью какой-либо осязаемый результат, необходимо сначала выполнить все скрипты, предназначенные для выгрузки.

Кнопка "Просмотр" генерирует таблицу по выбранному списку данных и фильтров (см. ниже). Таблица может быть отсортирована, распечатана или экспортирована в файл .csv обычным образом.

Фильтры. Фильтры сравнивают значения каждого из столбцов таблицы с заданными строками и числами. Фильтры по разным столбцам объединяются операторами AND (И) и OR (ИЛИ). В выгрузку включаются только те строки (клиенты), для которых результирующее логическое выражение принимает значение true. Клиенты, для которых оно принимает значение false, из выгрузки удаляются.

ПРИМЕЧАНИЕ Если требуются более сложные фильтры (например, сложные конструкции из AND и OR, или двусторонние ограничения вида min < ЗНАЧЕНИЕ < max), то следует написать их на странице "Групповые операции", по результатам создать там временную группу и в выгрузках просто указать эту группу.

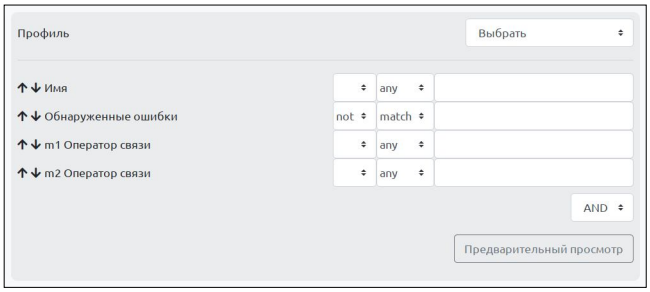
Для построения фильтров на данной странице имеются следующие элементарные действия:

any	Всегда возвращает true.
not any	Всегда возвращает false.
== < >	Сравнение чисел или строк (посимвольное, для кодировки UTF–8; например, ABC < XYZ есть true). Для == допускается только точное совпадение, с учётом регистра и без подстановочных символов (см. match). В частности, можно сравнивать строки с некоторой фиксированной датой/временем (записанной в формате ГГГГ-ММ-DD ЧЧ:ММ:СС), версией ПО и т.п.
match	Проверяет строку на соответствие точному образцу или шаблону. В шаблоне можно использовать следующие подстановочные символы и выражения: * любой один или несколько символов ? ровно один любой символ [abc] любой из перечисленных символов [a-z] любой из диапазона символов [!...] любой символ, кроме перечисленных
exists	Возвращает true, если строка является не пустой (эквивалентно not match + пустое последнее поле)
time_diff_gt time_diff_lt	Только для скриптов, возвращающих некоторую дату и время: вычислить разницу между ответом скрипта и текущей датой и сравнить её с заданной величиной. Параметр может указываться в секундах или в виде числа с одним из суффиксов s, m, h, d, например, 12h.

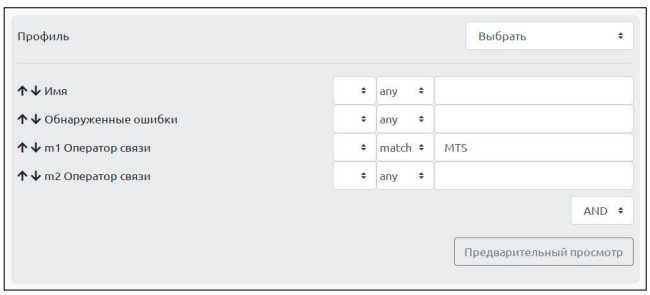
Примеры фильтров:

Вывести только клиентов NSG–1700:

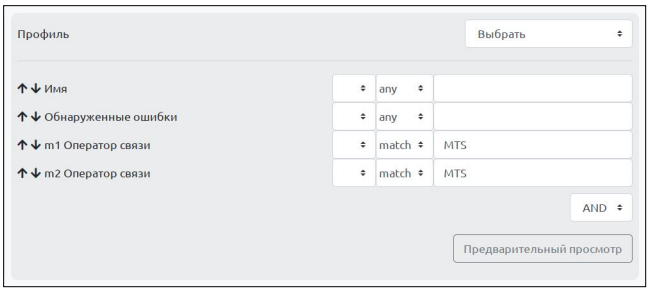
Вывести только клиентов, у которых имеются сообщения ИИ:



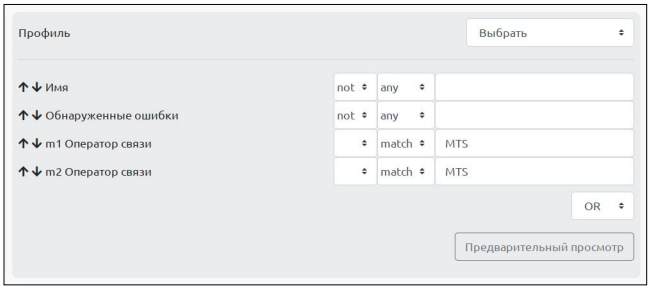
Вывести только клиентов, у которых в порту m1 установлена SIM-карта МТС:



Вывести только клиентов, у которых в обоих портах m1 и m2 установлены SIM-карты МТС:



Вывести только клиентов, у которых хотя бы в одном из портов m1 и m2 установлены SIM-карты МТС:

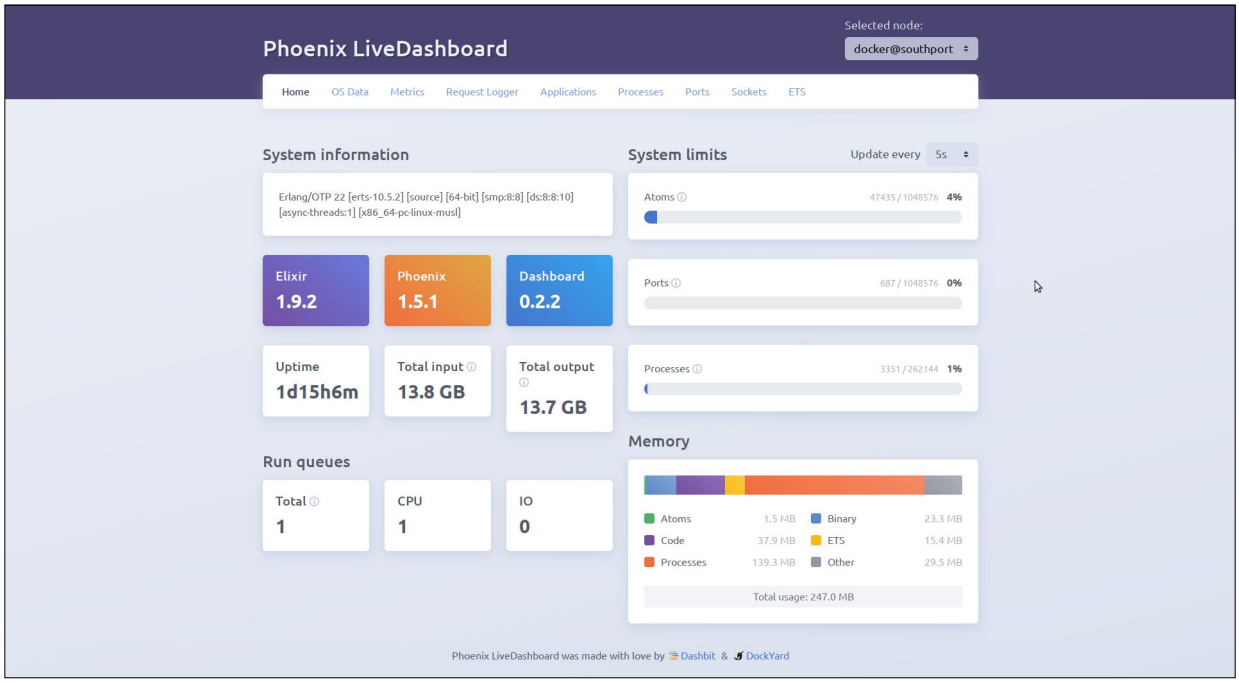


ПРИМЕЧАНИЕ Если используется оператор OR, то для полей, по которым фильтрация не требуется, надо ставить фильтр not any.

Профили. Кнопка 🖫 позволяет сохранить единожды составленный формат выгрузки в виде готового профиля, который можно будет использовать в будущем. Сохранённые ранее профили доступны в выпадающем списке вверху панели. Сохранённые профили выгрузок доступны всем пользователям.

§5.6. Системная панель

Системная панель предоставляет статистику системы Elixir на каждом сервере и инструменты для работы с ней. Наиболее существенная информация о загруженности системы, расходе вычислительных ресурсов, памяти, версиях отдельных компонент и т.п. приведена на первой вкладке. Панель открывается в отдельном окне или вкладке броузера.



§6. Шаблоны и скрипты

§6.1. Общие указания

Скрипты и шаблоны скриптов пишутся на различных языках, имеют различный формат и синтаксис в зависимости от модели оборудования, для которого они предназначены, её CLI и формата конфигурационных файлов. Именно в них содержится вся специфика конкретных типов используемого оборудования и их средств конфигурирования. В частности:

— Скрипты для клиентской и серверной стороны туннелей "Клещ" — на языке Elixir.

— Скрипты для устройств NSG и, в общем случае, для другого оборудования под управлением ОС Linux — на языке *bash* или иной командной оболочки Linux.

— Скрипты для устройств NSG могут содержать вызов из-под *bash* фирменной командной оболочки *nsgsh*. Рекомендуется по возможности избегать лишних вызовов через *nsgsh* и использовать напрямую утилиты NSG, для которых она является лишь пользовательским интерфейсом. Например, для обновления ПО устройства:

  nsgupdate -i -f *URL*

  Подробнее см. справку по NSG Linux 2.x на Web-сайте NSG

  http://www.nsg.ru/help

  или на борту устройства, раздел "Немного линукса".

— Неисполняемые конфигурационные файлы должны быть написаны в соответствии с синтаксисом соответствующего оборудования и ПО.

— Ключи и сертификаты, помещаемые на устройства, записываются в формате, который предусмотрен использующим их ПО (SSH и др.)

Исполняемые скрипты передаются на клиентское устройство в виде самораспакующегося архива, который содержит исполняемый файл и может содержать другие исполняемые и неисполняемые файлы, указанные в нём. Архив распаковывается во временную директорию, после чего исполняемый файл получает управление.

Во всех случаях в шаблонах могут (и в большинстве случаев должны, для этого они и предназначены) использоваться макроподстановки, с помощью которых определяются уникальные параметры для данного экземпляра туннеля. Макроподстановки задаются с помощью шаблонизатора Mustache:

http://mustache.github.io/mustache.5.html

Основные конструкции Mustache, используемые в шаблонах для систем "Богатка" и "Клещ":

{{client.params.*КЛЮЧ*}}

Параметр, вводимый или выбираемый вручную на странице настройки клиента (см. §3.7). Должен быть описан в поле "Определения параметров" класса, для которого предназначен данный шаблон (см. §3.5).

ПРИМЕЧАНИЕ В теле шаблонов пишется полное имя {{client.params.*КЛЮЧ*}}. В поле "Определения параметров" класса пишутся только имена ключей, в формате *КЛЮЧ : ЗНАЧЕНИЕ*. В результате на странице настройки клиента генерируются поля ввода или выбора, соответствующие каждому из *КЛЮЧЕЙ*.

{{*ПАРАМЕТР*}}

Параметр, вычисляемый автоматически. Должен быть определён в поле "Вычисление параметров" класса.

{{global.*КЛЮЧ*}}

Глобальный параметр, определённый на странице системных настроек (см. §3.11).

{{#fn.path_to}}*ФАЙЛ* {{/fn.path_to}}

Функция, выполняющая двоякую роль при упаковке и исполнении скрипта. Используется для копирования файлов с сервера на клиента:

— при упаковке архива включает в него указанный файл;

— при исполнении скрипта возвращает полный путь и имя файла во временной директории на клиенте.

Файл может быть как статическим, так и сгенерированным автоматически в процессе работы с данным клиентом (например, файл конфигурации, рассчитанный индивидуально для него).

{{#fn.include_templ}}*ФАЙЛ* {{/fn.include_templ}}

Функция, включающая содержимое указанного файла на сервере Богатка в тело скрипта (исполняемого или неисполняемого). Подробнее см. §6.9.

{{#fn.Lua}}*Lua_script* {{/fn.Lua}}

Функция на языке Lua. Mustache позиционируется как ""язык без логики" (logic-less): в нём нет никаких операторов условий или циклов. Если требуется что-то посложнее, чем простая подстановка, то данная конструкция позволяет вызвать для этого интерпретатор Lua. При этом тело секции является скриптом на Lua, а результат скрипта является собственно макроподстановкой.

Пример. Есть надобность иногда выключать сотовый порт, но не хочется плодить для этого отдельный параметр. Заодно хочется определять режимы работы в более удобных и коротких терминах, нежели этого требует язык конфигурации NSG. В совокупности, в классе определён следующий параметр для клиента:

```
"m1_mode":["auto","LTE/UMTS,"LTE","UMTS","down"]
```

а в конфигурации клиента применяем функцию на Lua следующим образом:

```
port
: m1
: : {{#fn.Lua}}
if "{{client.params.m1_mode}}"  == "down" then
  return 'adm-state = "down"'
elseif "{{client.params.m1_mode}}"  == "LTE" then
  return 'mode = "LTE only"'
elseif "{{client.params.m1_mode}}"  == "UMTS" then
  return 'mode = "UMTS only"'
else
  return 'mode = "{{client.params.m1_mode}}"'
end
{{/fn.Lua}}
```

(*пустая строка, поскольку один перенос строки после конца секции игнорируется*)

На выходе получается либо строка

```
: : adm-state = "down"
```

либо строка

```
: : mode = "..."
```

в соответствии с синтаксисом *nsgsh*. (Использовать обе эти строки одновременно нет смысла.)

Примеры наиболее важных типовых скриптов приведены ниже в данном разделе. (Не следует рассматривать этот набор и сами эти скрипты как окончательные и неизменные, они могут и будут модифицироваться в каждой конкретной инсталляции и по мере накопления опыта эксплуатации.)

§6.2. Шаблоны для туннелей "Клещ"

Для настройки туннелей "Клещ" необходимо сформировать файл конфигурации и передать его на клиента. Пример шаблона для клиента NSG, оборудованного двумя интерфейсами LTE — .acari_config_2lte.exs:

```
servers_list = [
[host: "my.poopy.host.1", port: 50019],
[host: "my.poopy.host.2", port: 50019]
]

# Линки
m1_link = [
dev: "m1",
table: 101,
servers: servers_list,
restart_script: "nsgsh -qr .port.m1.restart=nil"
]

m2_link = [
dev: "m2",
table: 102,
servers: servers_list,
restart_script: "nsgsh -qr .port.m2.restart=nil"
]

# Конфигурация клиента
# Должна быть последним выражением в этом файле
[
iface: [name: "acari0", addr: "{{tun_pref}}.{{hb}}.{{lb}}/32", peer: "{{tun_pref}}.0.1"],
links: [m1_link, m2_link]
]
```

Здесь:

servers_list Список серверов "Богатка/Клещ". 50019 — порт, назначенный для работы туннелей "Клещ" с шифрованием TLS (см. §2.4.). Для туннелей TCP без шифрования используется другой порт (по умолчанию 50020).

В описаниях каналов связи:

dev Системное имя интерфейса в терминах NSG Linux 2.x.

table Номер таблицы маршрутизации. Для доступа к разным серверам по разным каналам связи используются раздельные таблицы и маршрутизация на основе правил. Пользователю не надо об этом заботиться, необходимо только знать, что данные таблицы используются "Клещом" в случае, если ему понадобится добавлять какие-то свои таблицы и правила.

servers Список серверов, используемых для работы по данному каналу. В качестве значения можно указывать переменную (servers_list), определённую выше. Можно писать список напрямую, например:

```
m1_link = [
    servers: [
        [host: "my.poopy.host.1", port: 50019],
        [host: "my.poopy.host.2", port: 50019]
    ], ...
]
```

Такой формат удобен, если список относится только к одному линку (например,

данный оператор предоставляет услугу VPN, и адреса серверов в этой сети будут другими, чем в сетях общего пользования).
Число серверов в списках, относящихся к разным линкам, может не совпадать (если какие-то сервера по данному каналу связи не доступны).

restart_script
: Скрипт для рестарта (если это возможно) данного интерфейса при обнаружении его неработоспособности. Скрипт исполняется в контексте *bash*, но в данном случае управление сотовым модулем реализовано в *nsgconfd*, поэтому для передачи команды в него используется вызов *nsgsh* в пакетном режиме.
Для других типов и реализаций интерфейсов возможны, соответственно, другие варианты скрипта, как непосредственно в *bash*, так и через *nsgsh*.

В конфигурации клиента:

{{tun_pref}}
: Первые два байта пространства IP-адресов для данного класса клиентов — назначаются статически в описании класса.

{{hb}}, {{lb}}
: Третий и четвёртый байты IP-адреса для данного клиента — вычисляются динамически из серийного номера клиента. Правила вычисления заданы в описании класса.

Исходя из практического опыта, рекомендуется использовать для каждой модели клиентских устройств один из блоков адресов 172.16.0.0/16 ... 172.31.0.0/16. Это гарантирует уникальность IP-адресов для клиентов как одной и той же, так и разных моделей. (По крайней мере, пока их серийные номера не превышают 65535; после этого следует выделить новый блок адресов и скорректировать формулы для расчёта hb и lb.) При этом устройства одной модели могут входить в разные классы конфигурации.

На серверной стороне всех туннелей в данной схеме используется ненумерованный IP-интерфейс с адресом вида 172.x.0.1/32. (Предполагается, что устройства с серийным номером 000001 вряд ли попадут в руки заказчика, поэтому данный адрес свободен во всех блоках.)

Теоретически можно было бы использовать любой другой адресный план по усмотрению заказчика — например, назначить серверу адрес 10.0.0.1, клиенту 10.{{hb}}.{{lb}}.1 и выделить остаток блока 10.{{hb}}.{{lb}}.0/24 для подсети на удалённой площадке за клиентом. Но на практике адреса 10.0.0.0/8, равно как и 192.168.x.0/24, активно используются как в корпоративных сетях, так и в сетях поставщиков услуг. При совпадении адресов вероятны конфликты с маршрутизацией, которые придётся решать вручную и тем самым сводить на нет все достоинства автоматического назначения адресов. Либо использовать более сложную маршрутизацию с использованием правил. Блок 172.16.0.0/12 используется редко, и в этом отношении он наиболее безопасен.

Данный шаблон — простой текстовый файл, неисполняемый. Его необходимо поместить на клиентское устройство в строго определённое место и под определённым именем, зашитым в код клиента "Клещ":

```
#!/bin/bash
mkdir -p /etc/acari
cp {{#fn.path_to}}acari_config_2lte.exs{{/fn.path_to}} /etc/acari/acari_config.exs
```

Это уже будет частью исполняемого скрипта конфигурации (см. ниже).

На серверной стороне интерфейс для туннеля "Клещ" создаётся автоматически. Единственная операция, априори неизвестная при создании туннеля — это назначение IP-адресов концам туннеля. Только это и делает исполняемый скрипт для настройки туннеля — server_confg.sh:

```
#!/bin/bash
ip ad ad dev {{params.ifname}} {{tun_pref}}.0.1/32 peer {{tun_pref}}.{{hb}}.{{lb}}
```

Здесь {{params.ifname}} — имя данного туннельного интерфейса на серверной стороне, генерируется автоматически. (Полный список таких параметров можно вывести с помощью кнопки i на странице просмотра свойств шаблона.)

ПРИМЕЧАНИЕ Интерфейс на сервере с именем вида tunN создаётся при первом подключении клиента. Номер интерфейса — свой на каждом сервере, и в дальнейшем он не изменяется. Чтобы пересоздать интерфейсы для данного клиента на всех серверах, следует заблокировать и снова разблокировать его.

§6.3. Сторожевой таймер

Вспомогательный скрипт watchdog, который будет запускаться вообще вне всей системы конфигурации *nsgconfd*. Это дополнительная страховка на самый крайний случай. Он будет периодически посылать *ping* на сервер и при длительном отсутствии связи (инженерное правило здравого смысла: примерно в 3 раза больше типичного времени восстановления связи штатными средствами "Клеща") рестартовать устройство полностью, со сбросом заводской конфигурации.

```
#!/bin/bash
if [ "$1" = "DAEMON" ]; then
sleep 600
while ping -w600 -c1 {{tun_pref}}.0.1;
do
sleep 82
done
rm /etc/version
sync
reboot
fi
nohup setsid $0 DAEMON $* &>/dev/null &
```

Пока это неисполняемый текстовый файл. Единственный настраиваемый параметр, который в нём используется — это адрес серверной стороны туннеля для данного класса клиентов.

§6.4. Конфигурация клиента

Основная конфигурация клиента NSG хранится в отдельном файле nsg_config_2lte. Это неисполняемый текстовый файл. За рамками данной конфигурации остаётся туннель "Клещ", который не входит в общую систему конфигурации NSG Linux 2.*x*, имеет отдельный файл конфигурации и, как планируется, останется таковым в будущем. Единственный его след в конфигурации — это запуск его через services.daemons. Типовая конфигурация для устройства NSG–1700 с двумя модулями LTE:

```
ethernet
: bridge
: : br1
: : : ifAddress
: : : : prefix = "192.168.1.1/24"
ip
: route
: : 1
: : : network = "my.poopy.dns.server/32"
: : : device = "m1"
: : : metric = 100
: : 2
```

```
: : : network = "my.poopy.dns.server/32"
: : : device = "m2"
: : : metric = 200
: : 3
: : : network = "0.0.0.0/0"
: : : device = "acari0"
: nat
: : POSTROUTING
: : : 1
: : : : out-interface = "acari0"
: : : : target = "MASQUERADE"
: : : : _keep = true
port
: eth0
: : bridge-group = "br1"
: eth1
: : bridge-group = "br1"
: eth2
: : bridge-group = "br1"
: m1
: : type = "lte"
: : mode = "{{client.params.m1_mode}}"
: : debug-level=2
: : ifAddress
: : : dhcp-options
: : : : discard-default-gw = true
: : : : discard-dns = true
: : event-generator
: : : enable = true
: : : CSQ
: : : : sensor-name = "csq[m1]"
: : : : 1
: : : : : from = 99
: : : : : to = 99
: : : : : event-name = "-1"
: : : : : _keep = true
: : : : 2
: : : : : _keep = true
: m2
: : type = "lte"
: : mode = "{{client.params.m2_mode}}"
: : debug-level=2
: : ifAddress
: : : dhcp-options
: : : : discard-default-gw = true
: : : : discard-dns = true
: : provider
: : : main
: : : : attempts = 0
: : : aux
: : : : attempts = 1
: : event-generator
: : : enable = true
: : : CSQ
: : : : sensor-name = "csq[m2]"
```

```
: : : : 1
: : : : : from = 99
: : : : : to = 99
: : : : : event-name = "-1"
: : : : : _keep = true
: : : : 2
: : : : : _keep = true
services
: daemons
: : acari
: : : adm-state = "up"
: : : description = "acari client"
: : : command = "LC_ALL=ru_RU.utf8 /usr/lib/acari_{{arch}}/bin/acari_{{arch}} start"
: : : _keep = true
: dhcp
: : br1
: : : adm-state = "up"
: : : gateway = "192.168.1.1"
: : : ip-address-pool
: : : : from = "192.168.1.2"
: : : : to = "192.168.1.14"
: : : mask = "255.255.255.240"
: : : _keep = true
: event-handler
: : event-actions
: : : zabbixsender
: : : : acari_zbx
: : : : : hostname = "{{id}}"
: : : : : server = "{{tun_pref}}.0.1:50051"
: : : : : _keep = true
: : 1
: : : virt-sensor = "csq*"
: : : action = "#zabbixsender.acari_zbx"
: : : _keep = true
: : 2
: : : virt-sensor = "net*[m*]"
: : : action = "#zabbixsender.acari_zbx"
: : : _keep = true
: : 3
: : : virt-sensor = "net*[acari0]"
: : : action = "#zabbixsender.acari_zbx"
: : : _keep = true
: : 4
: : : virt-sensor = "net*[eth0]"
: : : action = "#zabbixsender.acari_zbx"
: : : _keep = true
: http
: : enable = false
: ssh
: : enable = true
: telnet
: : enable = false
system
: dns-client
: : name-server
```

```
: : : 1 = "my.poopy.dns.server "
: hostname = "{{id}}"
: ntp
: : enable = true
: timezone
: : location = "Moscow"
: users
: : root
: : : hash = "..."
```

Комментарии:

1. Для подключения к сетям общего пользования в данной конфигурации используются только сотовые интерфейсы. Все порты Ethernet объединены в один программный коммутатор (*bridge group*) для подключения одной лишь локальной сети.
2. Адреса DNS, назначаемые операторами — игнорируются. К сожалению, в настоящее время наблюдается практика назначения ими своих собственных DNS с приватными адресами, доступными только внутри их сетей. Это приводит к неработоспособной схеме, когда действующие DNS назначены от одного оператора, а приоритетный маршрут (с меньшей метрикой) указывает в сеть другого — и там DNS с таким адресом, естественно, нет. Поэтому вместо них следует жёстко назначить статические DNS и прописать явные маршруты на них напрямую через сети поставщиков услуг, минуя туннель (поскольку без них туннель и не поднимется). Рекомендуется использовать для этой цели сервера DNS самого заказчика, подконтрольные ему. DNS указываются по их реальным публичным адресам.
3. В данной конфигурации, со стороны туннелей и серверов "Богатка/Клещ" вся удалённая площадка представлена единственным IP-адресом туннельного интерфейса. Все исходящие соединения и, при необходимости, заданные входящие преобразуются NAT. Возможны и иные конфигурации.
4. Для портов LTE предусмотрены настраиваемые параметры m1_mode , m2_mode, соответственно. Эти параметры будут доступны на странице конфигурации клиента. На практике именно их приходится подбирать вручную наиболее часто при неудовлетворительном качестве сотовой связи. Показание CSQ=99 (нет сети) преобразуется в –1 (так удобнее с ним работать в Zabbix), остальные уровни сигнала передаются без изменений.
5. Все события с именами csq*, net*[m*], net*[acari0], т.е. показания соответствующих программных датчиков, передаются на порт Zabbix-прокси (50051) сервера "Богатка" (внутри туннеля).
6. В качестве *hostname* используется идентификатор устройства.
7. После завершения настройки и подтверждения работоспособности данной конфигурации её следует дополнить фильтрами в соответствии с корпоративной политикой безопасности.

§6.5. Ключ для SSH

Для входа на клиентское устройство по SSH рекомендуется использовать ключ RSA сервера (точнее, его открытую часть). Вход без ключа также возможен, но для этого придётся каждый раз вводить пароль, что не очень удобно.

Ссылка на ключ хранится в "Богатке" в неисполняемом текстовом файле id_rsa.pub :

```
{{#fn.include_file}}~/.ssh/id_rsa.pub{{/fn.include_file}}
```

а на клиента он передаётся, в качестве отдельной операции, исполняемым скриптом add_rsa_pub (который, в свою очередь, ссылается на вышеупомянутый):

```
#!/bin/bash
cat {{#fn.path_to}}id_rsa.pub{{/fn.path_to}} >> /root/.ssh/authorized_keys
```

Аналогичный фрагмент содержится в общем файле настройки клиента (см. ниже).

§6.6. Скрипты для настройки клиента

Скрипт .configure_2lte.sh выполняет несколько задач, описанных или упомянутых выше:

```
#!/bin/bash
mkdir -p /root/.ssh
cat {{#fn.path_to}}id_rsa.pub{{/fn.path_to}} >> /root/.ssh/authorized_keys
nsgsh -q "system.session.close=admin"
nsgconf2table {{#fn.path_to}}nsg_config_2lte{{/fn.path_to}} > /etc/nsgconfig
exit_status=$?
if [ $exit_status -ne 0 ]; then
echo "Ошибка при выполнении nsgconf2table: $exit_status"
exit $exit_status
fi

WD=/etc/private/init.d/watchdog.start
cp {{#fn.path_to}}watchdog{{/fn.path_to}} $WD
chmod a+x $WD

mkdir -p /etc/acari
cp {{#fn.path_to}}acari_config_2lte.exs{{/fn.path_to}} /etc/acari/acari_config.exs
echo "Новая конфигурация загружена."
echo "Клиент рестартует."
reboot
```

1. Создаёт на клиенте (на всякий случай, если нету) директорию для открытого ключа сервера и добавляет туда этот ключ.
2. Завершает (на всякий случай, если есть) работу администратора на данном клиенте. Конвертирует текстовый файл конфигурации из человеко-читаемого формата (с переносами строк, отступами и двоеточиями) в менее удобный формат, используемый *nsgconfd* (Для NSG Linix 2.1.3 и выше — не обязательно). Кладёт этот файл на клиента в /etc/nsgconfig.
3. Кладёт скрипт watchdog в директорию, предназначенную для пользовательских скриптов, которые надо запускать автоматически при старте системы. Назначает ему права исполняемого.
4. Создаёт (если нет) директорию для конфигурации "Клеща" и кладёт туда конфигурационный файл.

Аналогичный скрипт .config_update_2lte.sh предназначен для обновления конфигурации на работающем клиенте, и выполняет только п.2 из вышеприведённого списка:

```
#!/bin/bash
nsgsh -q "system.session.close=admin"
nsgconf2table {{#fn.path_to}}nsg_config_2lte{{/fn.path_to}} > /etc/nsgconfig
exit_status=$?
if [ $exit_status -ne 0 ]; then
echo "Ошибка при выполнении nsgconf2table: $exit_status"
exit $exit_status
fi

echo "Новая конфигурация загружена."
echo "Клиент рестартует."
reboot
```

§6.7. Общесистемные сервисные и информационные скрипты

Примеры разных полезных в жизни скриптов.

Опрос версии программного обеспечения (для устройств NSG Linux 2.x) — sw_version.sh:

```
#!/bin/bash
cat /etc/version
```

Опрос версии ПО "Клещ" — acari_version.sh . Получает номер версии и очищает ответ от ненужных элементов:

```
#!/bin/bash
wget -qO- --header "Content-Type: application/json" --post-data '{"fn":"get_version"}' \
http://localhost:50021/api | grep -oE '"result":"[0-9\.]*"' | grep -oE '[0-9\.]*'
```

Опрос времени непрерывной работы — uptime.sh. Выполняет команду *uptime*, извлекает из ответа время и преобразует его в единообразный формат, удобный для чтения и сравнения:

```
#!/bin/bash
uptime=$(cat /proc/uptime | grep -oE "^\s*[0-9]+")
printf '%4dd %02dh %02dm' $(($uptime / 86400)) $(($uptime % 86400 / 3600)) $(($uptime % 3600 / 60))
```

Рестарт клиента — reboot.sh:

```
#!/bin/bash
echo -n "Клиент будет рестартован через минуту: "
date -d @$((`date +%s` + 60))
echo "Остановить рестарт можно скриптом stop_reboot"
reboot -d60&
```

и отмена рестарта — stop_reboot.sh:

```
#!/bin/bash
killall -9 reboot
```

Вывод информации об аппаратной платформе (для устройств NSG Linux 2.x) — hw_info.sh:

```
#!/bin/bash
nsgsh -qr ".system.show.boardinfo"
nsgsh -qr ".system.show.cpuinfo"
```

Обновление ПО NSG Linux 2.x — full_update.sh :

```
#!/bin/bash
nsgupdate -if http://my.poopy.main.host/downloads/releases/nsg-client/latest/nsg-image-custom.bin
reboot -d5
echo OK
```

Обновление ПО "Клещ" — .acari_update.sh. ПО "Клещ" входит в состав ПО клиента и обновляется, как правило, вместе с ним. Но при необходимости оно может обновляться и отдельно. Скрипт загружает новую версию ПО "Клещ" с указанного URL, распаковывает её в /usr/lib , при успешном завершении передаёт текущему процессу "Клещ" через его WebAPI команду для завершения.

```
#!/bin/bash
TGZ_FILE=acari_{{arch}}.tar.gz
TMP_FILE=/tmp/$TGZ_FILE

rm -f $TMP_FILE
```

```
wget -nv http://my.poopy.main.host/downloads/releases/acari-client/latest/_FILE -O $TMP_FILE
if [ $? == 0 ]; then
tar -C /usr/lib -mxzf $TMP_FILE
if [ $? == 0 ]; then
wget -qO- --header "Content-Type: application/json" --post-data '{"fn":"halt"}' http://localhost:50021/api
echo OK
else
echo "Can't untar tarball"
fi
else
echo "Can't load tarball"
fi
rm -f $TMP_FILE
```

ПРИМЕЧАНИЕ В приведённой выше конфигурации для ПО NSG Linux 2.*x* запуск "Клеща" выполняется из узла services.daemons командного дерева. При этом демон конфигурации nsgconfd следит за состоянием этого процесса и в случае завершения немедленно рестартует его, уже из новой копии ПО. В других Linux-системах необходимо обеспечить перезапуск "Клеща" их специфическими средствами (например, с помощью *units* для *systemd*) или непосредственно в теле данного скрипта.

§6.8. Скрипты для контроля интерфейсов LTE

Примеры скриптов для опроса состояния модулей LTE. Скрипты выполняют команды show.module-info, show.radio-info, show.csq-check (в контексте оболочки *nsgsh*) и извлекают из ответов интересующую информацию.

Номер IMEI (NSG Linux 2.1.2 и ранее) — m1_IMEI.sh :

```
#!/bin/bash
nsgsh -iq "port.m1.show.module-info _quit" <&- 2>/dev/null | grep IMEI | tr -dc '0-9'
```

Аналогичным образом могут быть извлечены номер ICCID, производитель модуля, оператор, которому принадлежит SIM-карта и т.п. В версиях NSG Linux 2.1.3 и выше это уже не требуется, поскольку статические параметры модуля и подключения извлекаются автоматически при каждом переподключении и помещаются в файлы вида /var/log/mN/* .

Уровень сигнала сотовой сети — m1_csq.sh:

```
#!/bin/bash
nsgsh -q port.m1.show.csq-check <&- 2>/dev/null | tr -dc '0-9'
```

Температура модуля (для Sierra Wireless MC7304) — m1_temp.sh:

```
#!/bin/bash
nsgsh -iq "port.m1.show.radio-info _quit" <&- 2>/dev/null | grep -oE "Temperature.*" | tr -dc '0-9'
```

В отличие от IMEI, IMSI и других параметров, эти величины могут непрерывно меняться, и важно видеть их значения в текущий момент времени.

Если в клиентском устройстве установлено несколько модулей LTE, то эти скрипты можно тупо клонировать для каждого модуля. Более рациональный путь состоит в использовании вставок с помощью конструкции #fn.include_templ шаблонизатора (см. §6.9).

ПРИМЕЧАНИЕ Команды module-info, radio-info не предназначены для выполнения в пакетном режиме *nsgsh*. Чтобы обойти это ограничение, используется конструкция nsgsh -i ..... _quit <&- .

§6.9. Примеры однотипных скриптов с использованием вставок

Для написания похожих скриптов, различающихся небольшим числом параметров, целесообразно использовать конструкцию {{#fn.include_templ}} ... {{/fn.include_templ}}. Это позволяет сохранить всю содержательную часть скрипта в одном месте и не править её одинаково в нескольких местах, если в ней потребуются какие-либо изменения.

Пример: скрипты для опроса ICCID на сотовых интерфейсах (см. пред. параграф). Общая часть — шаблон ccid_inc:

```
nsgsh -iq "port.$PORT.show.module-info _quit" <&- 2>/dev/null | grep CCID | tr -dc '0-9'
```

Исполняемые скрипты для портов m1 и m2, соответственно:

```
#!/bin/bash
PORT=m1
{{#fn.include_templ}} ccid.inc {{/fn.include_templ}}
```

и

```
#!/bin/bash
PORT=m2
{{#fn.include_templ}} ccid.inc {{/fn.include_templ}}
```

Параметры, которые будут подставлены во включаемый шаблон, можно указать в самой функции include_templ. Например, шаблоны .acari_update_latest и .acari_update_stable отличаются одним словом. Можно написать общий шаблон acari_update.inc:

```
#!/bin/bash
TGZ_FILE=acari_{{arch}}.tar.gz
TMP_FILE=/tmp/$TGZ_FILE

rm -f $TMP_FILE
wget -nv http://10.45.32.32/download/releases/acari-client/{% type %}/$TGZ_FILE -O $TMP_FILE
if [ $? == 0 ]; then
  tar -C /usr/lib -mxzf $TMP_FILE
  if [ $? == 0 ]; then
    wget -qO- --header "Content-Type: application/json" --post-data '{"fn":"halt"}' http://localhost:50021/api
    echo OK
  else
    echo "Can't untar tarball"
  fi
else
  echo "Can't load tarball"
fi
rm -f $TMP_FILE
```

Тогда шаблоны .acari_update_latest и .acari_update_stable будут выглядеть так:

```
{{#fn.include_templ}} acari_update.inc {"type":"latest"} {{/fn.include_templ}}
{{#fn.include_templ}} acari_update.inc {"type":"stable"} {{/fn.include_templ}}
```

Параметры записываются в формате JSON, т.е. *КЛЮЧ* :*ЗНАЧЕНИЕ* .

Чтобы отличать глобальные подстановки (в данном примере {{arch}}) от подстановок, которые передаются в параметрах функции include_templ, последние заключаются в иной тег, по умолчанию — {% %}. В данном примере это {%type%}. Переопределить этот тег можно с помощью специального ключа "DELIMITER", например:

{{#fn.include_templ}} acari_update.inc {"type":"stable", "DELIMITER":"<% %>"} {{/fn.include_templ}}

§6.10.Скрипт для автоматического выбора SIM-карт

Устройства NSG–1700, за исключением ранних партий, оснащены 4 гнёздами для SIM-карт (по 2 на каждый сотовый интерфейс). Для единообразия с 2-симчатыми устройствами, на них используется та же самая конфигурация. Это требует, чтобы SIM-карты вставлялись в устройство строго определённым образом: в гнёзда m1 main и m2 aux.

Как показал опыт эксплуатации, квалификация выездных иженеров на местах не всегда достаточна для того, чтобы выполнить эту операцию без ошибок. Аналогичная проблема может иметь место с устройствами NSG–1750, а также с 2-симчатыми модификациями NSG–1700 и моделями NSG–1810/1820/1830, если они оснащены единственной сотовой опцией. Для решения этой проблемы предлагается следующий скрипт swap_restart типа Вставка.

Первое условие проверяет код продукта (E4ba1F7a0110 — 4-симчатая модификация). Во временном файле /tmp/sim.{%port%} сохраняется выбор карты для текущей попытки: 1 — main, 0 — aux. Если он существует, то это значение заменяет установки main.attempts, aux.attempts из конфигурации.

Второе условие проверяет наличие ошибки SIM в выводе команды show.progress. Если она есть, то карта меняется на противоположную, если нет — рестарт выполняется с этой же картой.

```
if [[ $(nsgsh -qr system.show.board | grep E4ba1F7a0110) ]]; then
 FILE="/tmp/sim.{%port%}";
 if [[ -e $FILE ]]; then
   MAIN_ATT=$(cat $FILE);
 else
   MAIN_ATT=$(nsgsh -qr port.{%port%}.provider.main.attempts._show | tr -dc '0-9');
 fi;

 if [[ $(nsgsh -qr port.{%port%}.show.progress | grep "ERR: SIM not inserted") ]]; then
   if [[ $MAIN_ATT > 0 ]]; then
     nsgsh -qf20 port.{%port%}.restart=aux;
     echo 0 > $FILE;
   else
     nsgsh -qf20 port.{%port%}.restart=main;
     echo 1 > $FILE;
   fi;
 else
   if [[ $MAIN_ATT > 0 ]]; then
     nsgsh -qf20 port.{%port%}.restart=main;
     echo 1 > $FILE;
   else
     nsgsh -qf20 port.{%port%}.restart=aux;
     echo 0 > $FILE;
   fi;
 fi;

else
  nsgsh -qf20 port.{%port%}.restart=nil;
fi;
```

Для 2-симчатой модификации рестарт выполняется обычным образом.

При рестарте устройства (может быть, его выключали именно для того, чтобы правильно вставить карту?) временный файл теряется и скрипт начинает работу снова, делая опять одну неправильную попытку.

Скрипт вызывается из основной конфигурации Клеща (acari_config_....exs), с указанием порта как параметра (см. §6.9):

```
.............................
m1_link = [
  .............................
  restart_script: ~S"""
{{#fn.include_templ}}swap_restart {"port":"m1"}{{/fn.include_templ}}
  """
]

m2_link = [
  .............................
  restart_script: ~S"""
{{#fn.include_templ}}swap_restart {"port":"m2"}{{/fn.include_templ}}
  """
]
.............................
```

Подробнее о *синтаксисе длинных строк* и о *sigils* в Elixir см.:

https://elixir-examples.github.io/examples/multiline-strings-heredocs
https://elixir-lang.org/getting-started/sigils.html

ПРИМЕЧАНИЕ Данная процедура актуальна для версий ПО 2.1.2.*x*. Для версий 2.1.3 и выше поиск SIM производится автоматически.

§6.11.Скрипты для Zabbix API

Данная категория скриптов представляет собой, по существу, дополнительные модули на языке Elixir. Текст, вычисленный из шаблона, на лету компилируется в функцию и эта функция выполняется в среде сервера "Богатка", как отдельный процесс. Такой механизм позволяет гибко модифицировать возможности системы в части вывода и анализа статистики и подстраивать их под потребности заказчика.

Для написания шаблонов Zabbix API используются:

— Язык Elixir:
https://hexdocs.pm/elixir/Kernel.html
— Zabbix API:
https://www.zabbix.com/documentation/current/ru/manual/api
— Реализация Zabbix API в "Богатке"
описание в разработке

По состоянию на текущий момент, скрипты Zabbix API разрабатываются в порядке авторского сопровождения.

§7. Подробно о настройке туннелей Клещ

§7.1. Синтаксис файла конфигурации клиента

Конфигурация клиента по существу представляет собой текст на языке Elixir. Формальное описание файла:

```
КОНФИГУРАЦИЯ := [ links: [ КАНАЛ, ... ], iface: ИНТЕРФЕЙС ]

КАНАЛ :=  [ dev: ИМЯ,
            table: ТАБЛИЦА_МАРШРУТИЗАЦИИ,
            restart_script: СКРИПТ,
            gw: ШЛЮЗ,
            servers: [СЕРВЕР, ... ]
          ]

ИНТЕРФЕЙС :=  [ name: ИМЯ,
                addr: IP-ПРЕФИКС,
                peer: IP-АДРЕС,
                broadcast: ШИРОКОВЕЩАТЕЛЬНЫЙ_АДРЕС,
                tap: true | false,
                proto: "ssl" | "tcp"
              ]
```

Квадратные скобки здесь — это обычный символ, а не метасимвол "условное вхождение" EBNF. Выражения в квадратных скобках являются списками. Запятые после каждого элемента списка, кроме последнего — обязательны, после последнего — допустимы.

Каждый из ключей (приведённых строчными буквами) заканчивается двоеточием. Пробел между ключом и двоеточием недопустим, после двоеточия — обязателен. Иначе говоря, двоеточие есть обязательная часть имени ключа. Правильный синтаксис:

```
[key: value]
```

Ошибочный синтаксис:

```
[key : value]
[key:value]
```

Остальные пробелы и переносы строк опциональны.

Строки, начинающиеся с решётки, являются комментариями.

Теперь в переводе на язык, приближенный к человеческому:

Конфигурация клиента состоит из описаний одного или нескольких каналов связи и одного описания интерфейса.

iface: Описание интерфейса. Ключи описания интерфейса:

- **tap:** Тип интерфейса:
  - false — интерфейс типа tun (Layer 3), по умолчанию
  - true — интерфейс типа tap (Layer 2).
- **name:** Имя интерфейса, строка.
- **addr:** IP-префикс интерфейса, строка ("a.b.c.d/m"). Для интерфейсов типа tun — обязательный параметр.
- **peer:** IP-адрес удалённой стороны, строка.
- **broadcast:** Широковещательный адрес, строка.

Значения addr, peer, broadcast, в конечном счёте, подставляются в команду

```
ip addr add dev <name> <addr> peer <peer> broadcast <broadcast>
```

Если имя интерфейса не задано, оно будет назначено автоматически в формате tunN или tapN, соответственно.

links: Список каналов связи. Каждый элемент списка есть описание конкретного канала. Ключи описания канала:

dev: Имя интерфейса, строка. Параметр обязательный.

ВНИМАНИЕ Параметр dev необходим не только для маршрутизации. По этому имени Клещ определяет интерфейс, соответствующий выбранному каналу связи, его IP-адрес, принудительно подставляет этот адрес в исходящий пакет как адрес источника, и только затем вбрасывает пакет в подсистему маршрутизации. Маршрутизация выполняется на основе правил, критерием в которых является адрес источника.

table: Номер таблицы маршрутизации, число от 1 до 252. Подробнее см. §7.3.

restart_script:
Скрипт для рестарта канала, строка. Зависит от типа интерфейса и возможностей управления им в данном устройстве.

gw: Шлюз, может быть строкой или функцией (см. ниже).

proto: Протокол соединения, строка:
"tls" — TLS v1.3, по умолчанию
"tcp" — чистое TCP-соединение без шифрования
Если указано любое другое значение, используется TLS.

servers: Список серверов. Ключи описания сервера:
host: Адрес сервера, строка.
port: Порт, число.

ПРИМЕЧАНИЕ Для соединений с шифрованием и без используются разные порты на сервере, заданные в скрипте run-bogatka. (По умолчанию, 50019 и 50020, соответственно.)

tls_options: Настройки шифрования TLS. Ключи описания сервера:
versions: Версия TLS.
ciphers: Описание алгоритмов аутентификации, шифрования и проверки целостности (в формате функции ssl.connect языка Erlang).
Пример:

```
tls_options: [
  versions: [:"tlsv1.2"],
  ciphers: [{:dhe_rsa, :aes_128_cbc, :sha}]
]
```

Если опции заданы неверно и попытка установить их завершается с ошибкой, то повторный вызов идет с настройками TLS по умолчанию.

ПРИМЕЧАНИЕ Набор поддерживаемых алгоритмов определяется при компиляции ПО. Чтобы просмотреть список в формате, указанном выше, следует выполнить в консоли сервера Богатка (т.е. внутри контейнера) команду:
/opt/app/bin/bogatka_docker rpc ":ssl.cipher_suites(:all) |> IO.inspect(limit: :infinity)"
а на клиенте — команду:
/usr/lib/acari_APX/bin/acari_APX rpc ":ssl.cipher_suites(:all) |> IO.inspect(limit: :infinity)"
где APX — архитектура данной модели клиентов: arm, arm64 либо powerpc.

Функции. В описании канала связи в качестве значения параметра gw можно использовать функцию, если нельзя задать этот параметр статически. Эта функция будет вызываться каждый раз, когда системе нужно получить этот параметр. Значение функции будет являться значением параметра. Функция — это анонимная функция, на языке Elixir.

Пример конфигурации. Предполагается, что адрес шлюза поставщика услуг, полученный при конфигурации, не устанавливается в качестве шлюза по умолчанию, а записывается в файл /var/dhcp/eth1/discardrouter (почему — см. подробнее §7.3 о маршрутизации):

```
[
  iface: [name: "acari0", addr: "172.16.13.91/32", peer: "172.16.0.1"],
  links: [
    [
      dev: "eth1",
      gw: fn ->
            case File.read("/var/dhcp/eth1/discardrouter") do
              {:ok, content} -> content
              _ -> nil
            end
          end,
      table: 100,
      servers: [
        [host: "1.2.3.4", port: 50019],
        [host: "5.6.7.8", port: 50019]
      ]
    ],
    [
      dev: "m1",
      table: 101,
      servers: [
        [host: "1.2.3.4", port: 50019],
        [host: "5.6.7.8", port: 50019]
      ],
      restart_script: "nsgsh -qr .port.m1.restart=nil"
    ],
  ]
]
```

Переменные. Для удобства написания конфигурации, можно использовать переменные. Переменным необходимо сначала присвоить части конфигурации, а затем использовать их ниже по тексту. Пример — та же самая конфигурация:

```
# Список серверов
servers = [
  [host: "1.2.3.4", port: 50019],
  [host: "5.6.7.8", port: 50019]
]

# Функция получения шлюза
get_gw = fn ->
  case File.read("/var/dhcp/eth1/discardrouter") do
    {:ok, content} -> content
    _ -> nil
  end
end
```

```
# Каналы связи
# Используют переменные servers и get_gw, поэтому должны быть описаны ниже них
eth1_link = [
  dev: "eth1",
  gw: get_gw,
  table: 100,
  servers: servers
]

m1_link = [
  dev: "m1",
  table: 101,
  servers: servers,
  restart_script: "nsgsh -qr .port.m1.restart=nil"
]

# Конфигурация клиента
# Должна быть в этом случае последним выражением в файле
[
  iface: [name: "acari0", addr: "172.16.13.91/32", peer: "172.16.0.1"],
  links: [eth1_link, m1_link]
]
```

§7.2. Конфигурация серверной стороны

Конфигурация серверной стороны, в простейшем случае, состоит в настройке адресов для IP–интерфейса сервера. Все туннели между сервером и клиентами являются соединениями "точка-точка", поэтому удобно использовать для сервера ненумерованный интерфейс с одним и тем же "одолженным" адресом, а для клиента — вычислять 2 байта адреса автоматически из его заводского номера. Это, однако, является не более чем рекомендацией, пользователь вправе использовать другие варианты распределения адресов по своему усмотрению.

```
#!/bin/bash
ip addr add dev {{params.ifname}} {{tun_pref}}.0.1/32 peer {{tun_pref}}.{{hb}}.{{lb}}
```

Интерфейс на сервере создаётся при первом подключении клиента. Имя интерфейса генерируется автоматически в формате tunN или tapN, в зависимости от типа интерфейса, настроенного на стороне клиента. Номер интерфейса — свой на каждом сервере, и в дальнейшем он не изменяется.

Если клиент заблокирован, то этот интерфейс удаляется и создаётся заново при разблокировке; его номер при этом может быть передан какому-нибудь другому новому интерфейсу. Чтобы пересоздать интерфейсы для данного клиента на всех серверах, следует заблокировать и снова разблокировать его. Это следует сделать, например, если клиент перенесён в другой класс с другой схемой назначения IP-адресов.

Для туннеля типа tap вышеуказанная конфигурация также применима и, более того, маршрут в сеть peer/32 также корректен. Но практического смысла такая конфигурация не имеет, она приводит только к лишним накладным расходам на передачу заголовка Ethernet в каждом пакете. Туннели типа tap предназначены для того, чтобы прозрачно пробросить, например, ПК удалённого сотрудника в сеть офиса. В этом случае следует не назначать им IP-адреса, а включить их в *bridge group*:

```
#!/bin/bash
brctl addif br1 {{params.ifname}}
```

(Предполагается, что группа br1 к этому моменту уже создана и в неё изначально включена ЛС офиса.)

§7.3. Маршрутизация на клиенте

Получение маршрутов. Все интерфейсы современных клиентских устройств (Ethernet, LTE, WiFi) являются формально широковещательными. Соответственно, в описаниях маршрутов необходимо указывать IP-адреса ближайших шлюзов. По смыслу задачи, эти адреса получаются на каждом интерфейсе динамически по протоколу DHCP.

Для интерфейсов LTE маршрутизацию можно несколько упростить, указав в маршруте вместо шлюза (gw) выходной интерфейс (dev). Это формально некорректно, но де-факто модули LTE позволяют работать в таком режиме, отвечая на широковещательные пакеты, которые приходят в их внутренний порт.

Для интерфейсов Ethernet и WiFi такая подмена уже не работает, необходимо честно указывать полученный адрес шлюза. Для этого требуется отдельный механизм, который позволит каким-то образом определить адрес шлюза и динамически внести его в конфигурацию Клеща. Задача усложняется тем, что его требуется не использовать в качестве шлюза по умолчанию (см. ниже) — а, соответственно, необходимо его хранить где-то отдельно.

В устройствах NSG для этой цели предусмотрена следующая процедура. Все полученные параметры DHCP сохраняются в файлах /var/dhcp/ИНТЕРФЕЙС/*. В Клеще имеется функция get_gw, которая каждый раз считывает адрес шлюза из файла /var/dhcp/ИНТЕРФЕЙС/discardrouter и подставляет его в описание канала через этот ИНТЕРФЕЙС (см. §7.1).

В других системах эту задачу следует решать другими доступными методами, например, с помощью скриптов ip-up и/или настроек клиента DHCP.

Маршрутизация. По постановке задачи, маршрутизацию на клиенте требуется организовать следующим образом:

— Весь трафик с удалённой площадки и с самого устройства, адресованный в корпоративную сеть, должен направляться через туннель.
— Через публичные интерфейсы и сети поставщиков услуг должны отправляться только запросы DNS (чтобы разрешить адреса серверов) и пакеты самого туннеля Клещ.
— Пакеты Клещ на каждый сервер должны отправляться через все публичные интерфейсы.

Такая маршрутизация не реализуется с помощью одной таблицы. Она требует наличия нескольких таблиц и правил (политик) маршрутизации.

— Маршруты на сервера DNS указываются статически, через все публичные интерфейсы. Маршрутам назначаются разные метрики, чтобы они не конфликтовали в таблице маршрутизации.
— Маршрут по умолчанию назначается через интерфейс Клеща.
— Маршруты по умолчанию, назначаемые поставщиками услуг, игнорируются. (Но сами адреса шлюзов при этом сохраняются отдельно.)
— Также игнорируются их динамически назначаемые DNS, вместо него используются статические DNS, желательно собственные корпоративные.

```
ip
: route
: : 1
: : : network = "my.poopy.dns.server/32"
: : : device = "m1"
: : : metric = 100
: : 2
: : : network = "my.poopy.dns.server/32"
: : : device = "m2"
: : : metric = 200
: : 3
: : : network = "0.0.0.0/0"
: : : device = "acari0"
```

```
: nat
: : POSTROUTING
: : : 1
: : : : out-interface = "acari0"
: : : : target = "MASQUERADE"
: : : : _keep = true
port
: m1
: : type = "lte"
: : ifAddress
: : : dhcp-options
: : : : discard-default-gw = true
: : : : discard-dns = true
: m2
: : type = "lte"
: : ifAddress
: : : dhcp-options
: : : : discard-default-gw = true
: : : : discard-dns = true
system
: dns-client
: : name-server
: : : 1 = "my.poopy.dns.server"
```

— Для каждого публичного интерфейса используется своя таблица маршрутизации. Номера таблиц указываются в описании канала связи в файле конфигурации Клеща (см. §7.1).

— Клиент Клещ принудительно формирует пакеты с разными адресами источника (взятыми от публичных интерфейсов).

— Правила маршрутизации определяют, что пакет с адресом источника, взятым от интерфейса X, должен маршрутизироваться по таблице, указанной в описании этого канала связи X.

Эти таблицы и правила создаются Клещом автоматически. В конфигурации устройства ничего для этого делать не надо. Пользователю, который составляет или отлаживает конфигурацию, следует лишь иметь в виду их наличие и избегать конфликтов с собственными таблицами и правилами, если они ему понадобятся. (Пример см. в §7.4.) Ознакомиться с ними можно при помощи команд ip route show table HOMEP, ip rules show (в *bash*) или ip.route-tables.HOMEP.show, ip.rules.show (в *nsgsh*). Пример:

```
# ifconfig
m1        .....
          inet addr:10.160.25.73  P-t-P:10.160.25.73  Mask:255.255.255.252
m2        .....
          inet addr:10.224.140.46  P-t-P:10.224.140.46  Mask:255.255.255.252
.....

# ip rule show
0:        from all lookup local
32755:    from 10.160.25.73 lookup 101
32756:    from 10.224.140.46 lookup 102
32766:    from all lookup main
32767:    from all lookup main

# ip route show table 101
A.B.C.D dev m1 scope link
W.X.Y.Z dev m1 scope link

# ip route show table 102
A.B.C.D dev m2 scope link
W.X.Y.Z dev m2 scope link
```

§7.4. Маршрутизация при конфликтующих адресах

В современной практике поставщики услуг Интернет размещают клиентов в своих приватных сетях, использующих адресные пространства 10.0.0.0/8, 192.168.0.0/16 и, существенно реже (но также не исключено) 172.16.0.0/12. Эти же адреса используются и в корпоративных сетях. Не исключена ситуация, когда адреса, назначенные поставщиком услуг, совпадут с адресами каких-либо нужных хостов в сети заказчика. Последствия этого труднопредсказуемы, но почти неизбежно нарушат штатную работу клиента.

Для избежания такой ситуации приходится существенно усложнить конфигурацию, используя маршрутизацию на основе установленных правил (политик). Маршрут по умолчанию в туннель Клещ указывается в отдельной таблице маршрутизации. Все пакеты, входящие в маршрутизатор через локальный интерфейс (в примере ниже — br1) или формально входящие через локальный интерфейс lo, маршрутизируются отдельными правилами по этой таблице.

Пример. Интерфейсы (лишний вывод опущен):

```
# ifconfig
acari0  .....
        inet addr:172.16.13.79  P-t-P:172.16.0.1  Mask:255.255.255.255
m1      .....
        inet addr:10.66.175.167  P-t-P:10.66.175.167  Mask:255.255.255.240
m2      .....
        inet addr:172.30.8.185  P-t-P:172.30.8.185  Mask:255.255.255.252
```

Как видим, сотовый оператор на порту m2 использует адреса вида 172.x.x.x. Хотя в данном случае они и не совпадают с адресом, назначенным Богаткой интерфейсу acari0, сам факт двойного использования этого диапазона вынуждает принять предупредительные меры.

Фрагмент конфигурации устройства NSG. Для записи маршрута в заданную таблицу используется штатный механизм с помощью демона динамической маршрутизации BIRD.

```
# nsgsh -qr _print
ip
: route
  <маршрута по умолчанию нет>
: route-tables
: : 1
: : : _keep              = true
: dynamic-routing
: : enable               = true
: : kernel
: : : 1
: : : : export           = "all"
: : : : kernel-table      = 1
: : static
: : : 1
: : : : route
: : : : : 0.0.0.0/0
: : : : : : action       = "via-interface"
: : : : : : interface     = "acari0"
: rule
: : 32764
: : : iif              = "br1"
: : : table              = 1
: : 32765
: : : iif              = "lo"
: : : table              = 1
```

```
port
: m1
: : type                = "lte"
: : ifAddress
: : : dhcp-options
: : : : discard-default-gw = true
: m2
: : type                = "lte"
: : ifAddress
: : : dhcp-options
: : : : discard-default-gw = true

.............................
```

Правила маршрутизации. Правила 32762–32763 созданы Клещом, таблицы 101–102 указаны в его конфигурации:

```
# ip rule sh
0:      from all lookup local
32762:  from 172.30.8.185 lookup 102
32763:  from 10.66.175.167 lookup 101
32764:  from all iif br1 lookup 1
32765:  from all iif lo lookup 1
32766:  from all lookup main
32767:  from all lookup main
```

Таблицы маршрутизации:

```
# ip route sh table 1
default dev acari0 proto bird

# ip route show table 101
A.B.C.D dev m1 scope link
W.X.Y.Z dev m1 scope link

# ip route show table 102
A.B.C.D dev m2 scope link
W.X.Y.Z dev m2 scope link

# ip route sh — здесь только маршруты в непосредственно подключённые сети
10.66.175.160/28 dev m1 proto kernel scope link src 10.66.175.167
172.16.0.1 dev acari0 proto kernel scope link src 172.16.13.79
172.30.8.184/30 dev m2 proto kernel scope link src 172.30.8.185
```

ВНИМАНИЕ Правило `from all iif lo lookup 1` в данной конфигурации определяет маршрутизацию для пакетов, исходящих с данного устройства в корпоративную сеть. (Телеметрия, SSH и т.п.) Оно должно следовать после правил для маршрутизации пакетов Клещ (т.е. иметь бóльший номер).
Поскольку номера для правил Клеща выбираются динамически, то данное правило следует создавать под последним доступным номером — 32675. Тогда правила Клеща гарантированно окажутся впереди него.
Контроль за непротиворечивостью правил и их использованием целиком возлагается на пользователя.

ПРИМЕЧАНИЕ Приведённый выше набор правил не позволяет зайти на устройство из локальной сети, поскольку ответы будут отправляться не обратно в неё, а в туннель. Но в данной задаче доступ из локальной сети и не предполагается, клиентское устройство управляется только из центра.