

# Руководство администратора

# платформы "Центральный Пульт"

v.3.2.76 - 25.01.2019

# Содержание

1. Описание документа	2
2. Назначение системы	4
3. Системные требования	6
3.1 Системные требования к серверу .....	7
3.2 Системные требования к агенту .....	8
4. Принципы функционирования системы	9
4.1 Архитектура системы .....	10
4.2 Описание функционирования системы и её частей .....	12
5. Обязанности и задачи администратора	14
5.1 Настройка параметров работы системы .....	15
5.2 Управление логированием .....	31
5.3 Управление учётными записями пользователей .....	36
5.4 Работа с объектами и связями .....	38
5.5 Настройка уведомлений .....	43
5.6 Настройка интерфейса .....	46
5.7 Настройка мониторинга .....	49
5.8 Ипорт и экспорт данных .....	52
6. Проблемы в работе системы и способы их решения	53
6.1 Недостаточно места на виртуальной машине с сервером .....	54
6.2 Отсутствие подключения агента к серверу .....	55
6.3 Проверка работы MongoDB .....	56
6.4 Проверка работы MySQL .....	56
6.5 Проверка работы Redis .....	57
6.6 500 Internal server error и отсутствие графиков .....	58
6.7 Ошибка работы HTTP-проверки .....	58

# **Описание документа**

**ЦЕНТРАЛЬНЫЙ ПУЛЬТ**

# Описание документа

## 1      Описание документа

Настоящий документ является руководством для администрирования автоматизированной системы "Центральный пульт" и предназначен для конкретизации задач и функций должностных лиц организации (предприятия, фирмы), планирующих и осуществляющих сбор, хранение, передачу и анализ данных по объектам мониторинга с применением системы.

В документе приведены основные функции администратора, архитектура системы и её модулей, алгоритм создания учётных записей, порядок установки прав доступа пользователей и другие сведения необходимые для управления АС "Центральный пульт".

# **Назначение системы**

**ЦЕНТРАЛЬНЫЙ ПУЛЬТ**

# Назначение системы

## 2 Назначение системы

Система предназначена для визуализации и мониторинга различных объектов. "Центральный пульт" нацелен на упрощение сбора данных, ускорение их анализа, визуализации результатов и беспрерывного хранения.

Автоматизации подвергаются следующие функциональные возможности процесса мониторинга:

- процесс обработки данных;
- хранение оригинальных значений;
- обеспечение анализа информации;
- управление объектами мониторинга;
- уведомление пользователей о состояниях объектов;
- исправление аварийных ситуаций;
- преобразование данных в компактный вид;
- экспорт данных;
- удаление устаревших данных.

# **Системные требования**

**ЦЕНТРАЛЬНЫЙ ПУЛЬТ**

# Системные требования

## 3 Системные требования

Система может быть установлена на выделенных аппаратных или виртуальных мощностях.

### 3.1 Системные требования к серверу

Для работы сервера системы требуется следующая конфигурация:

- 64-bit OS
- CPU - 4 cores
- RAM - 8 GB
- HDD - 72 GB

Для надёжной работы сервера рекомендуются операционные системы:

- Ubuntu Linux 14.04 / 16.04;
- Ubuntu Linux 10.04 / 12.04;
- Red Hat Enterprise Linux 5.5+ / 6 / 7;
- SUSE Linux Enterprise 11 / 12.

Под сетевой конфигурацией понимаются открытые порты:

- 80 - HTTP
- 443 - HTTPS
- 8091 - web socket
- 6379 - REDIS Server
- 1162 - SNMP catcher@SAYMON Agent
- 22 - SSH
- 1883/8883 — MQTT/MQTT

Серверная часть системы может быть поставлена в виде готового образа виртуальной машины или Docker-контейнера.

Объём образа виртуальной машины составляет 5 GB и доступен для скачивания: [yadi.sk/d/EKWcd89n3NsADX](https://yadi.sk/d/EKWcd89n3NsADX).

# Системные требования

Стандартные логин / пароль - saymon / saymon.

Содержимое скачанного архива состоит из нескольких компонентов:

- Ubuntu-14.04.5-server-amd64;
- SAYMON Server 2.0.67;
- SAYMON Agent.

Серверная часть в виде Docker-контейнера может быть поставлена на следующих ОС:

- CentOS / Debian / Fedora / Ubuntu Linux
- macOS;
- Windows 10.

Подробное описание установки представлено на сайте: [www.docker.com](http://www.docker.com).

## 3.2 Системные требования к агенту

Агенты обладают кросс-платформенной совместимостью и могут быть установлены на различные операционные системы:

- Ubuntu Linux;
- Red Hat Enterprise Linux / CentOS Linux;
- Raspberry Pi;
- Mac OS X;
- Windows.

Требованием к операционным системам является поддержка Java SE 6, 7 и 8.

Рекомендуемая конфигурация для работы агентов системы:

- OS with Java 6/7/8 support
- CPU — 2 GHz single core;
- RAM — 1 GB;
- HDD — OS + 2 GB.

# **Принципы функционирования системы**

**ЦЕНТРАЛЬНЫЙ ПУЛЬТ**

# Принципы функционирования системы

## 4 Принципы функционирования системы

### 4.1 Архитектура системы

Платформа имеет клиент-серверную архитектуру и включает в себя три основных уровня:

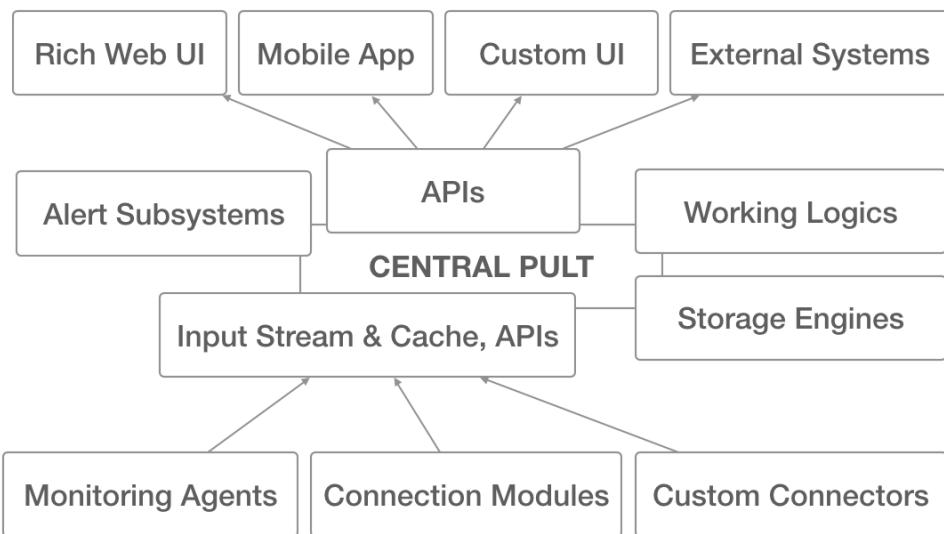


Рисунок.1. Общая архитектура платформы "Центральный Пульт"

Нижний уровень предназначен для сбора данных и осуществления управления над полученной информацией, которая периодически отправляется в Input Stream & Cache для дальнейшей обработки и анализа. Данный слой включает в себя три элемента:

- Monitoring Agents - агенты системы, собирающие с узлов мониторинга информацию.
- Connection Modules - готовое и переиспользуемое решение для сбора информации без агента.
- Custom Connectors - разработанные для клиента интерфейсы, осуществляющие мониторинг объектов без агента.

На втором уровне хранится и обрабатывается полученная от агентов и прочих интерфейсов информация, а затем передаётся клиентам в требуемом для каждого из решений виде.

Второй слой подразделяется на:

# Принципы функционирования системы

- Working Logics (бизнес-логики) - совокупность правил, принципов, зависимостей поведения объектов, на основе которых обрабатываются поступившие с нижнего слоя данные и сохраняются в Storage Engines для анализа ситуаций в настоящем и прошлом и построения математически обоснованных прогнозов в будущем.
- Storage Engines - отвечает за хранение данных. Storage Engines реализовано в виде:
  - SQL - формирует запросы, описывающиеся, какую информацию из Storage Engines необходимо получить, пути решения определяются автоматически;
  - noSQL - обеспечивает гибкость и согласованность системы, благодаря гарантированному завершению запроса;
  - TimeSeries - специализированный компонент управления базой данных временных рядов, что позволяет хранить данные с высокой скважностью.
- Alert Subsystems - каналы уведомлений, по которым осуществляется информирование пользователей о смене состояний объектов, разрыве соединений и других нестандартных ситуациях. При соответствующей настройке "Центральный Пульт" может отправлять email-уведомления, SMS, сообщения в Telegram, совершать голосовые вызовы и отображать визуальные уведомления в браузере, сопровождающиеся звуком.
- APIs - относятся к категории REpresentational State Transfer (REST), что позволяет выполнять RESTful-операции на добавление, чтение, изменение и удаление информации для облачной учетной записи или инсталляции на сервере.
- Input Stream & Cache - хранилище данных, поступивших в сервер платформы с нижнего уровня, и их хранение. Cache реализован в виде сетевого журналируемого хранилища Redis. Взаимодействие между агентами и сервером платформы осуществляется при помощи Kafka: собирает данные с ниже располагающегося слоя, хранит данные у себя в распределённом хранилище по топикам и передаёт их серверу по запросу.

# Принципы функционирования системы

Верхний уровень отвечает за визуализацию полученных и обработанных данных, а также осуществление операций над ними конечным пользователем. В качестве средства интеграции приложений используются открытые API-интерфейсы.

- Rich Web UI - веб-интерфейс платформы, является основным средством работы с системой для конечного пользователя, где при наличии определённых прав возможны изменения как всей структура, так и отдельного объекта.
- Mobile App - мобильные приложения для операционных систем Android и iOS.
- Custom UI - кастомизированные интерфейсы, созданные под специальную бизнес-задачу или проект. Функциональные возможности позволяют вносить изменения - добавлять, удалять и редактировать - объекты аналогично Rich Web UI. Веб-интерфейс имеет уникальное отображение, согласно заявленным требованиям.
- External Systems - внешние для отображения или сбора данных системы, полученные средствами мониторинга платформы "Центральный Пульт".

## 4.2 Описание функционирования системы и её частей

Программное обеспечение платформы "Центральный Пульт" имеет открытые API-интерфейсы, которые обеспечивают информационную совместимость системы и возможность интеграции с другими автоматизированными системами.

С помощью API в веб-интерфейсе системы возможно реализовать большинство операций, некоторые из них:

- получение списка объектов;
- получение текущего статуса объектов и истории их состояний;
- запись данных в объекты без использования агентов;
- изменение свойств объектов;
- работа с инцидентами;

# Принципы функционирования системы

- создание классов с добавлением проверок.

Система состоит из следующих логических подсистем:

- Server - централизованный сервер, на котором хранится и анализируется информация, полученная от агентов, а затем отдаётся Клиенту.
- СУБД (MongoDB, OpenTSDB) - совокупность программных средств, предназначенных для создания, использования и управления базами данных.
- Agent - множество агентов системы, установленных на узлах инфраструктуры и собирающих информацию по ним. Полученные данные периодически отправляются в кэш и затем анализируются сервером.
- Клиент - тонкий веб-клиент системы и клиенты для мобильных платформ Android и iOS.

# **Обязанности и задачи администратора**

**ЦЕНТРАЛЬНЫЙ ПУЛЬТ**

# Обязанности и задачи администратора

## 5      Обязанности и задачи администратора

### 5.1    Настройка параметров работы системы

Раздел настройки - один из самых важных разделов, предназначенный для администратора системы. Этот раздел наполнен советами о том, как настроить "Центральный пульт" для мониторинга вашей среды, начиная настройкой сервера для получения необходимой информации и заканчивая просмотром данных, настройкой оповещений и удаленных команд, выполняемых в случае возникновения проблем.

#### 5.1.1    Установка агента

Перед началом работы с платформой "Центральный пульт" необходимо выполнить следующие действия:

- зайти на сайт платформы "Центральный пульт" [www.saymon.info/podderzhka/downloads](http://www.saymon.info/podderzhka/downloads) и скачать подходящую под ОС сборку агента. Или перенести с CD-ROM, USB-накопителя или другого носителя, на котором поставляется платформа "Центральный пульт", подходящую под пользовательскую ОС сборку агента на компьютер, сервер или устройство, на котором планируется осуществлять сбор данных;
- продолжить установку и настройку агента.

##### 5.1.1.1    Автоинсталляция

Чтобы приступить к началу работы и настройке мониторинга, необходимо выполнить следующие действия:

1. Открыть веб-интерфейс платформы "Центральный пульт".
2. Ввести логин и пароль учётной записи.  
*Примечание: возможно изменение пароля, для этого следует:*
  - нажать на логин в панели инструментов;
  - в выпадающем меню выбрать "Конфигурация";
  - в разделе "Смена пароля" ввести текущий пароль, новый;
  - подтвердить новый пароль, нажать "Сохранить".

# Обязанности и задачи администратора

3. Создать объект класса "Saymon agent", для этого нажать кнопку

**+ Создать объект** ("Создать объект") на панели инструментов:

3.1. Навести на созданный объект указатель мыши и нажать на появившуюся кнопку ("Настройки агента").

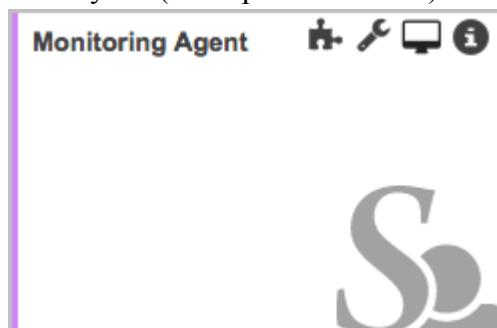


Рисунок 2

3.2. В появившемся окне скопировать ссылку из строки "Команда для установки агента".

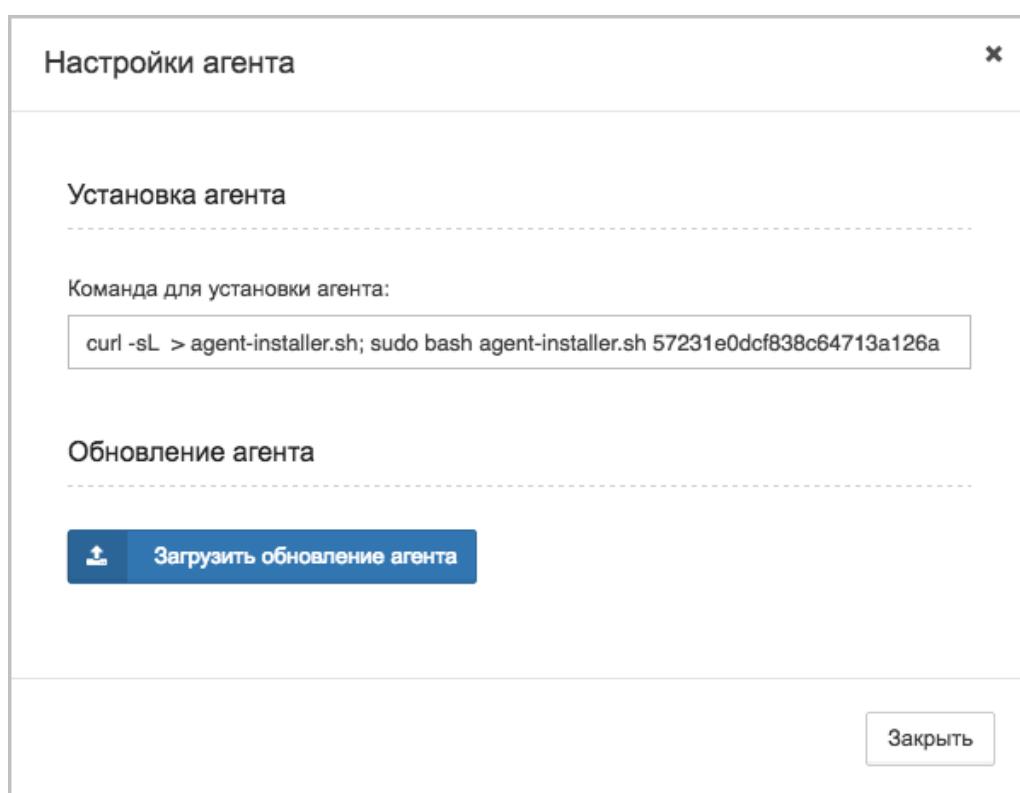
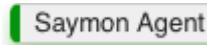


Рисунок 3

3.3 В терминале выполнить данную команду на необходимом сервере.

*Примечание: номер, который дал сервер - последние 24 символа команды, совпадает с ID агента.*

# Обязанности и задачи администратора

3.4. В течение 30 секунд агент скачивается и устанавливается. Если установка произведена корректно, состояние агента должно быть отражено зелёным цветом 

## 5.1.1.2 Ubuntu Linux

Чтобы приступить к началу работы и настройке мониторинга, необходимо выполнить следующие действия:

1. Создать системного пользователя, из-под которого будет запускаться агент:

```
sudo adduser --system --no-create-home saymon
```

2. Скачать архив Linux 64 bit или Linux 32 bit в папку */opt*.

3. Распаковать его:

```
sudo tar zxf saymon-agent-rl-linux-xXX-jre.tar.gz && sudo rm saymon-agent-rl-linux-xXX-jre.tar.gz
```

4. Назначить владельцем директории пользователя SAYMON agent:

```
saymon sudo chown -R saymon:saymon saymon-agent
```

5. Отредактировать файл настроек *saymon-agent/conf/agent.properties* (см. п. 3.4. "Конфигурация агента").

6. При необходимости включить централизованное логирование агента через SYSLOG в "saymon-agent/conf/logback-upstart.xml".

7. Установить скрипты для автоматического запуска агента:

### Ubuntu 14.04, Upstart

```
sudo cp saymon-agent/upstart/saymon-agent.conf /etc/init/ && sudo initctl reload-configuration
```

### Ubuntu 16.04, Systemd

```
sudo cp saymon-agent/systemd/saymon-agent.service /etc/systemd/system && sudo systemctl enable saymon-agent
```

8. Запустить сервис агента:

```
sudo start saymon-agent
```

## 5.1.1.3 Red Hat Enterprise Linux / CentOS Linux

Чтобы приступить к началу работы и настройке мониторинга, необходимо выполнить следующие действия:

# Обязанности и задачи администратора

1. Создать системного пользователя, из-под которого будет запускаться агент:

```
sudo useradd -r saymon
```

2. Скачать архив CentOS/RedHat v4 64 bit или CentOS/RedHat v4 32 bit в папку */opt*.

3. Распаковать его:

```
sudo tar zxf saymon-agent-rl-java6-linux-XXX-jre.tar.gz && sudo rm saymon-
agent-rl-java6-linux-XXX-jre.tar.gz
```

4. Назначить владельцем директории пользователя SAYMON agent.:

```
sudo chown -R saymonsaymon saymon-agent
```

5. Отредактировать файл настроек *saymon-agent/conf/agent.properties* (см. п. 3.4. "Конфигурация агента").

6. При необходимости включить централизованное логирование агента через SYSLOG в "saymon-agent/conf/logback-upstart.xml".

7. Установить "init.d" скрипты для автоматического запуска агента:

```
sudo cp saymon-agent/init.d/saymon-agent /etc/init.d/ && sudo chkconfig --add
saymon-agent
```

8. Запустить сервис агента:

```
sudo service saymon-agent start
```

## 5.1.1.4 Raspberry Pi (Raspbian OS)

Чтобы приступить к началу работы и настройке мониторинга, необходимо выполнить следующие действия:

1. Обновить репозиторий пакетов и системное ПО:

```
sudo apt-get update && sudo apt-get upgrade
```

2. Создать системного пользователя, из-под которого будет запускаться агент:

```
sudo adduser --system --no-create-home --group saymon
```

3. Установить Upstart и подтвердить замену sysvinit:

```
sudo apt-get install upstart
```

4. Перезагрузить систему, чтобы изменения вступили в силу:

# Обязанности и задачи администратора

```
sudo reboot now
```

5. Установить Oracle JDK 7:

```
sudo apt-get install oracle-java7-jdk
```

6. Убедиться, что команда `java-version` выводит корректную версию JRE.

7. Скачать ([www.saymon.info/podderzhka/downloads](http://www.saymon.info/podderzhka/downloads) или с CD-ROM, USB-накопителя иди другого носителя, на котором поставляется платформа "Центральный пульт") архив "`saymon-agent-VERSION-linux-generic.tar.gz`" в папку `"/opt"`, если ранее это не было сделано.

8. Распаковать его:

```
sudo tar zxf saymon-agent-VERSION-linux-generic.tar.gz && sudo rm saymon-
agent-VERSION-linux-generic.tar.gz
```

9. Назначить владельцем директории пользователя SAYMON agent:

```
sudo chown -R saymonsaymon saymon-agent
```

10. Отредактировать файл настроек `saymon-agent/conf/agent.properties` (см. п. 3.4. "Конфигурация агента").

11. При необходимости включить централизованное логирование агента через SYSLOG в `saymon-agent/conf/logback-upstart.xml`.

12. Установить upstart-скрипты для автоматического запуска агента:

```
sudo cp saymon-agent/upstart/saymon-agent.conf /etc/init/ && sudo initctl reload-
configuration
```

13. Запустить сервис агента:

```
sudo start saymon-agent
```

## 5.1.1.5 Прочие дистрибутивы Linux

Чтобы приступить к началу работы и настройке мониторинга, необходимо выполнить следующие действия:

1. Создать системного пользователя SAYMON AGENT, из-под которого будет запускаться агент.

2. Скачать архив Linux 64 bit или Linux 32 bit в папку `/opt`.

3. Распаковать его:

# Обязанности и задачи администратора

```
sudo tar zxf saymon-agent-rl-linux-xXX-jre.tar.gz && sudo rm saymon-agent-rl-
linux-xXX-jre.tar.gz
```

4. Назначить владельцем директории пользователя SAYMON agent:

```
sudo chown -R saymon:saymon saymon-agent
```

5. Отредактировать файл настроек *saymon-agent/conf/agent.properties* (см. п. 3.4. "Конфигурация агента").

6. Создать папку для хранения лог файлов:

```
sudo mkdir saymon-agent/log && sudo chown saymon:saymon saymon-agent/log
```

7. При необходимости включить централизованное логирование агента через SYSLOG в *saymon-agent/conf/logback.xml*.

8. Запустить агента:

```
cd saymon-agent && sudo -u saymon saymon-agent.sh
```

## 5.1.1.6 Mac OS X

Чтобы приступить к началу работы и настройке мониторинга, необходимо выполнить следующие действия:

1. Создать системного пользователя "Saymon agent", из-под которого будет запускаться агент.

2. Скачать архив ([www.saymon.info/podderzhka/downloads](http://www.saymon.info/podderzhka/downloads) или с CD-ROM, USB-накопителя или другого носителя, на котором поставляется платформа "Центральный пульт") "Mac OS X 64 bit" и распаковать его в папку */opt*, если ранее это не было сделано.

3. Отредактировать файл настроек *saymon-agent/conf/agent.properties* (см. п. 3.4. "Конфигурация агента").

4. Создать папку для хранения лог файлов:

```
sudo mkdir saymon-agent/log && sudo chown -R saymon:staff saymon-agent
```

5. При необходимости включить централизованное логирование агента через SYSLOG в *saymon-agent/conf/logback.xml*.

6. Добавить права на запуск агента:

```
sudo chmod +x saymon-agent/saymon-agent.sh
```

7. Запустить агента:

# Обязанности и задачи администратора

```
cd saymon-agent && sudo -u saymon ./saymon-agent.sh
```

## 5.1.1.7 Windows

Чтобы приступить к началу работы и настройке мониторинга, необходимо выполнить следующие действия:

1. Скачать ([www.saymon.info/podderzhka/downloads](http://www.saymon.info/podderzhka/downloads) или с CD-ROM, USB-накопителя или другого носителя, на котором поставляется платформа "Центральный пульт") инсталлятор "Windows 64 bit" или "Windows 32 bit", если ранее это не было сделано.
2. Открыть папку, в которой находится скачанный файл, и запустить его от имени администратора.
3. Выбрать место расположения папки с установкой (например, *C:\Program Files (x86)\SAYMON Agent*) и нажать на кнопку "Install" ("Установить").
4. Когда установка будет завершена, нажать на кнопку "Close" ("Закрыть").
5. Зайти в папку установки агента (например, *C:\Program Files (x86)\SAYMON Agent\conf*) и открыть файл "Agent".
6. Отредактировать файл в соответствии с конфигурацией агента (см. п. 3.4. "Конфигурация агента").
7. Сохранить файл: Файл - Сохранить.
8. Запустить службу "SAYMON Agent" в меню: Пуск - Панель управления - Администрирование - Службы.
9. После запуска службы "SAYMON Agent" зайти на сайт компании облачной платформы "SAYMON": [saas.saymon.info](http://saas.saymon.info), где настройка мониторинга станет возможна.

## 5.1.2 Конфигурация агента

Конфигурационный файл агента *../saymon-agent/conf/agent.properties* содержит следующие параметры:

### Общие настройки:

# Обязанности и задачи администратора

- agent.id= - уникальный идентификатор объекта "SAYMON Agent" в веб-интерфейсе системы;
- agent.activeMode=true - включает активный режим агента, снижающий задержки в отправке информации на сервер, но генерирующий дополнительный трафик.
- agent.maxParallelTasks - максимальное количество одновременно выполняемых агентом задач (0 или отрицательное значение - без ограничений);
- agent.discoveryEnabled - включает автоматический поиск сетевых устройств, выполняемый данным агентом (предварительно необходимо задать параметр "discovery\_parent\_id" в секции "server" конфигурационного файла .../etc/saymon/saymon-server.conf).

## Настройки сервера:

- server.host= - адрес сервера системы;
- server.port= - порт Redis, по которому осуществляется подключение агентов к серверу;
- server.password= - пароль Redis;
- server.maxConnections - максимальное количество одновременных подключений агента к серверу (минимум 2).

## Настройки получения SNMP Trap:

- agent.snmpTrapEnabled= - позволяет использовать агента в качестве получателя трапов;
- agent.snmpTrapListenPort= - порт для получения трапов;
- agent.snmpTrapReceiverThreadPoolSize= - количество одновременных потоков для получения трапов.

### Пример

```
#  
# SAYMON Agent configuration file.  
#  
# Agent ID.  
#
```

# Обязанности и задачи администратора

## Пример

```
# 1. Find a host in the model, which corresponds to this host, where the agent
#   is being installed.
# 2. Find a SAYMON Agent object on this host (if not present - create it).
# 3. Specify the ID of a SAYMON Agent object as a value of this property.
#
# Value 0 means agent will attempt to register on server automatically
# and receive it's ID during handshake.
#
agent.id=596f08ac3e6df9131c38ed81

# Enables/disables active mode.
#
# In active mode the agent will notify the data server with additional message on
# each new sensor reading. This allows for faster system reaction to state changes,
# but introduces some additional traffic. Disabling active mode on some agents
# allows to unload the server in case this is needed.
agent.activeMode=true

# SAYMON Server connection parameters.
server.host=saas.saymon.info
server.port=6379
server.password=easy1234
server.maxConnections=2

# SNMP trap settings.
agent.snmpTrapEnabled=false
agent.snmpTrapListenPort=1162
agent.snmpTrapReceiverThreadPoolSize=4

# Maximum number of parallel tasks (0 or negative means unlimited).
agent.maxParallelTasks=0

# Discovery settings.
agent.discoveryEnabled=true
agent.discoveryEnabled=false
```

### 5.1.3 Конфигурация сервера

Конфигурация сервера системы выполняется в файле:

*/etc/saymon/saymon-server.conf*

Далее приведены описания разделов и настроек.

В разделе *cache {}* задается пароль для доступа к серверу "Redis":

# Обязанности и задачи администратора

```
"auth_pass": "пароль"
```

В разделе `mqtt` {} задаются настройки подключения к MQTT-брокеру:

```
"broker": "адрес и порт брокера", например "mqtt://localhost:1883"
```

*Примечание: При использовании SSL-шифрования необходимо добавить пути к сертификату удостоверяющего центра, к ключу и сертификату клиента следующим образом:*

```
"ca": "path/to/ca.server.crt",
"certificate": "path/to/client.crt",
"key": "path/to/client.key".
```

В разделе `openTsdb` {} задаются параметры доступа к OpenTSDB:

```
"enabled": "запись исторических данных в OpenTSDB (false или true, по умолчанию true)",
"host": "адрес хоста с OpenTSDB",
"port": "порт OpenTSDB"
```

В разделе `server` {} задаются общие параметры сервера:

```
"sql_history_enabled": "запись исторических данных в MySQL (false или true, по умолчанию false)",
"history_update_period": "интервал записи исторических данных в миллисекундах; 0 - немедленная запись пришедших значений",
"redis_populate_period": "интервал записи данных об объектах и связях (ID) из MongoDB в Redis (в миллисекундах)",
"default_result_timeout": "время, через которое срабатывает условие "нет данных" с момента создания объекта или получения последних данных от агента (в миллисекундах)",
"default_state_id": "состояние объекта при создании (по умолчанию 7 - "нет данных"),
"debug": "debug-режим для логирования в saymon-server.log (false или true, по умолчанию false)",
```

# Обязанности и задачи администратора

```

" sms_script": "путь до скрипта, отправляющего sms-уведомления",
" voice_call_script": "путь до скрипта, осуществляющего голосовой вызов",
" comet_port": "порт для защищенного соединения",
" comet_secure": "ssl-соединение (false или true, по умолчанию false)",
" comet_ssl_certificate": "путь к сертификату",
" comet_ssl_key": "путь к ключу",
" self_object_id": "ID объекта, используемого для самомониторинга",
" colorize_log": "цветная раскраска лога (false или true, по умолчанию
false)",
" event_log_max_bytes": "размер записей консоли в mongoDB в байтах, при
достижении которого происходит ротация данных (по умолчанию 1
GB)",
" notification_buffering_period": "период ожидания для сбора сообщений о
смене состояний объектов и отправки пользователю сгруппированного
уведомления (в миллисекундах)",
" conditional_incidents_enabled": "включает функционал инцидентов (false
или true, по умолчанию false)",
" discovery_parent_id": "id объекта, в котором появляются найденные
агентами сетевые устройства",
" retain_expired_stat": "хранение последних полученных данных после их
устаревания (false или true, по умолчанию false)",
" stat_local_timestamp_field_name": "имя поля, в котором передаётся время, с
которым нужно сохранять данные в OpenTSDB"

```

В подразделе *email {}* задаются параметры доступа к почтовому серверу:

```

" disables": "отправка почтовых уведомлений (false или true, по
умолчанию true),
" transport" {}
    " service": "почтовый сервер",
    " auth" {}
        " user": "логин пользователя для доступа к почтовому серверу",
        " pass": "пароль пользователя для доступа к почтовому серверу",

```

# Обязанности и задачи администратора

```

“fields” {}

“from” : “почтовый адрес отправителя, например,
saymon@saas.saymon.info”,

“max_json_length”: “ограничение размера письма с уведомлением в
символах (если не указано - 1000 по умолчанию)”

```

В разделе *rest\_server* {} задаются параметры REST-сервера:

```

“ip_address” : “адрес хоста для запуска REST-сервера”,

“port” : “порт REST-сервера”,

“base_url” : “путь к API (по умолчанию `/api` )”,

“debug” : “режим debug для логирования в saymon-rest-server.log (false или
true, по умолчанию false)”,

“document_storage_dir” : “директория для хранения .pdf, .jpeg и других
файлов документации, прикрепляемых к объектам”,

“document_download_url” : “url к сохраненным в “$document_storage_dir
файлам” (по умолчанию `http://localhost/node/api/docs` )”,

“update_download_url” : “путь к файлу для обновления агента (по
умолчанию `http://localhost/node/api/agents/update` )”,

“colorize_log” : “цветная раскраска лога (false или true, по умолчанию
false)”,

“public_url” : “адрес для доступа к WEB-интерфейсу”

```

В разделе *ldap* {} задаются параметры внешнего LDAP-сервера для авторизации пользователей:

```

“url” : например, “ldap://127.0.0.1:389”,
“suffix” : например, “dc=saymon,dc=info”,
“login” : например, “cn=admin,dc=saymon,dc=info”,
“pass” : например, “root”

```

Для применения изменений необходимо перезапустить сервис saymon-server:

# **Обязанности и задачи администратора**

```
sudo service saymon-server restart
```

Управление некоторыми настройками доступно в веб-интерфейсе системы. Измененные параметры в Web UI приоритетнее настроек в конфигурационном файле. Установленные значения параметров автоматически сохраняются в MongoDB, файл `/etc/saymon/saymon-server.conf` не перезаписывается.

Окно "Опции конфигурации" состоит из двух разделов:

1. Общие настройки - соответствуют параметрам раздела *server* {}:

**Общие настройки**

---

Сохранять устаревшие данные от агента  НЕТ

Период обновления истории, мс

Период обновления таблицы состояний, мс

Период проверки устаревания данных от агента, мс

Период обработки пассивных данных от агента, мс

Максимальный размер журнала событий, байты

Срок действия данных от агента по умолчанию, мс

*Рисунок.4. Общие настройки сервера*

2. Настройки электронной почты - соответствуют параметрам раздела *email* {}:

# Обязанности и задачи администратора

*Рисунок.5. Настройки электронной почты*

Просмотреть актуальные настройки сервера возможно при помощи REST API-методов:

```
/node/api/configuration
```

## 5.1.4 Перезапуск сервера

1. Открыть терминал на хосте с сервером SAYMON.
2. Выполнить команду

```
sudo service saymon-server restart
```

Подтверждение успешного перезапуска выглядит следующим образом:

```
saymon-server stop/waiting  
saymon-server start/running
```

## 5.1.5 Просмотр websocket-нотификации

"Центральный пульт" использует гибкий механизм оповещений, который позволяет пользователю оперативно реагировать на возникшие нестандартные ситуации.

Чтобы просмотреть все пришедшие веб-сокет уведомления, необходимо:

1. Запустить Chrome;
2. Перейти по ссылке [saas.saymon.info](http://saas.saymon.info);
3. Ввести логин и пароль для авторизации;
4. Перейти по ссылке [saas.saymon.info/incidents.html?debug=comet](http://saas.saymon.info/incidents.html?debug=comet);
5. Открыть в Chrome Инструменты разработчика (Меню - Дополнительные инструменты - Инструменты разработчика);
6. Открыть вкладку Консоль (Console) в Инструментах разработчика;

# Обязанности и задачи администратора

7. В качестве примера нажать правой кнопкой мыши на строке с инцидентом и выбрать пункт подтвердить;
8. В консоли отобразится websocket-нотификация.

## 5.1.6 Увеличение количества обработчиков SNMP-Trap

Для поддержания бесперебойного наблюдения за объектами и оповещения администратора "Центральный пульт" использует SNMP-Trap.

Если на сервер поступает большое количество SNMP-Trap, то существует способ увеличения количества их обработчиков. Для этого необходимо:

1. Открыть в текстовом редакторе файл настроек акторов:

```
sudo nano /usr/local/saymon/backend/server/actors.json
```

2. Добавить следующую секцию:

```
"snmpTrapMessageHandlerActor": {
    "mode": "forked",
    "clusterSize": 3,
    "onCrash": "respawn"
}
```

3. Перезапустить сервер:

```
sudo service saymon-server restart
```

Для проверки корректности выполненной операции выполнить команду:

```
ps ax | grep node
```

В выводе должно отображаться то количество процессов SnmpTrapMessageHandlerActor, которое было указано в параметре "clusterSize": 3 выше:

# Обязанности и задачи администратора

```
5793 ? Rsl 10:23 /usr/bin/nodejs --
harmony /usr/local/saymon/backend/server/saymon-server.js

5857 ? S1 3:18 /opt/nodejs/bin/node --
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js
RestServerActor

5862 ? S1 0:37 /opt/nodejs/bin/node --
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js
ResourceServerActor

5867 ? S1 1:43 /opt/nodejs/bin/node --
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js
HistoryWriterActor

5872 ? S1 0:45 /opt/nodejs/bin/node --
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js
SnmpTrapMessageHandlerActor

5877 ? S1 0:42 /opt/nodejs/bin/node --
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js
SnmpTrapMessageHandlerActor

5882 ? S1 0:42 /opt/nodejs/bin/node --
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js
SnmpTrapMessageHandlerActor
```

## 5.1.7 Сброс системы к заводским настройкам

Пользователь может в любой момент сбросить настройки "Центрального пульта", чтобы восстановить настройки по умолчанию.

Для этого необходимо выполнить следующий скрипт:

```
#!/bin/bash
#
# Script deletes instance-specific data from MongoDB, Redis and OpenTSDB.
# Lets stop SAYMON Server and begin.
service saymon-server stop

mongo saymon --eval 'db.dropDatabase()'

echo flushall | redis-cli

docker stop opentsdb
docker rm opentsdb
```

# Обязанности и задачи администратора

```
docker run -d -p 127.0.0.1:4242:4242 --restart=always --name=opentsdb  
rossinno/opentsdb  
  
# Purges logs also.  
rm /var/log/saymon/* /var/log/nginx/saymon*  
  
# Forward to new monitoring adventures!  
service saymon-server start
```

## 5.2 Управление логированием

Для хранения и дальнейшего анализа изменений в системной работе агента "Центральный пульт" использует логирование.

Логи агента как правило хранятся в папках:

- /var/log/saymon
- /var/log/upstart/
- /opt/saymon-agent/log
- C:\Program Files\SAYMON Agent\log

### 5.2.1 Конфигурация log-файлов

Конфигурация log-файлов агента осуществляется в файле:

*/opt/saymon-agent/conf/logback-upstart.xml*

Секция настройки debug-режима:

# Обязанности и задачи администратора

```
<appender name="FILE-DEBUG"
  class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>/var/log/saymon/saymon-agent.debug.log</file>
  <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <!-- Daily rollover -->
    <fileNamePattern>/var/log/saymon/saymon-agent.debug.%d{yyyy-MM-
dd}.log.gz</fileNamePattern>
    <!-- Keep 10 days' worth of history -->
    <maxHistory>10</maxHistory>
  </rollingPolicy>
  <encoder>
    <pattern>%d{dd.MM.yyyy HH:mm:ss.SSS} [%-15thread] %-5level %
logger{36} - %msg%n</pattern>
  </encoder>
</appender>
```

где:

<file>/var/log/saymon/saymon-agent.debug.log</file> – размещение log-файла;  
<fileNamePattern>... .gz</fileNamePattern> – указание на архивацию файлов в формат .gz;  
<maxHistory>10</maxHistory> – длительность хранения файлов в днях.  
Для отключения debug-режима закомментировать соответствующую строку в секции ROOT следующим образом:  
<!--<appender-ref ref="FILE-DEBUG"/>-->

Настройки и структура секции базового логирования аналогична секции настройки debug-режима:

```
<appender name="FILE-INFO"
  class="ch.qos.logback.core.rolling.RollingFileAppender">
  ...
</appender>
```

## 5.2.2 Конфигурация ротации log-файлов

Конфигурация ротации log-файлов сервера выполняется в файле:  
`/etc/logrotate.d/saymon`

в котором:

`/var/log/saymon/saymon-rest-server.log /var/log/saymon/saymon-server.log` - размещение log-файлов;  
`daily` - ежедневная ротация;  
`missingok` - продолжать ротацию без ошибки, если отсутствует один из

# Обязанности и задачи администратора

файлов;  
 rotate N - длительность хранения файлов в днях;  
 compress - архивация файлов в формат .gzip;  
 notifempty - не производить ротацию лога, если он пуст;  
 copytruncate - писать лог в один файл, урезая его после каждого шага ротации.

## 5.2.3 Просмотр информации о журнале событий

Информацию о содержании журнала событий возможно просмотреть с помощью REST API метода:

```
/node/api/event-log/info
```

## 5.2.4 Назначение ответственного за событие

При возникновении критической ситуации возможно установить ответственного за неё пользователя двумя способами:

1. Через Web UI:

1.1. В панели режимов отображения переключиться на Журнал событий.



1.2. Нажать правой кнопкой мыши на требуемое событие, затем на "Назначить ответственного".

1.3. Выбрать ответственного за событие из выпадающего списка.

STAGING Журнал Событий								
Количество	Время	Критичность	Объект на схеме	Адрес отправителя	OID трапа	Текст	Данные	Ответственный
100	19.06.2018, 17:20:46	Major	SNMP TRAP	127.0.0.1	.1.3.6.1.4.1.5089.2.0.99	"TEST TEST"	.1.3.6.1.4.1.5089.2.0.99 "TEST TEST"	demo

Рисунок.6. Журнал событий

2. REST API методом:

```
/node/api/event-log/:id/assignee
```

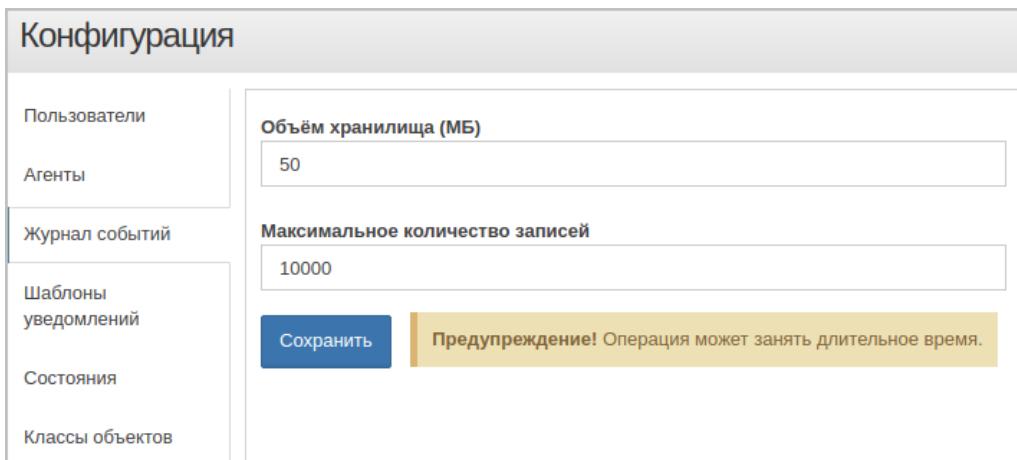
## 5.2.5 Установление ограничения для логирования

"Центральный пульт" позволяет установить максимальный объём хранилища двумя способами:

1. Через Web UI:

# Обязанности и задачи администратора

- 1.1. В панели инструментов нажать на имя пользователя и в выпадающем меню выбрать пункт "Конфигурация".
- 1.2. Перейти в раздел "Журнал событий".
- 1.3. Заполнить поля требуемыми значениями.
- 1.4. Сохранить изменения.



*Рисунок.7. Ограничение объёма хранилища*

2. Через REST API метод:

```
/node/api/event-log/limits
```

## 5.2.6 Удаление snmp-трапов из EventLog

Для очищения EventLog в веб-интерфейсе необходимо:

1. Выполнить в терминале следующие команды:

```
mongo saymon
>db.eventLog.drop()
```

2. После сброса требуется проверить создание новой коллекции командой:

```
>db.eventLog.stats()
```

3. Если ответ выглядит следующим образом:

```
{ "ok" : 0, "errmsg" : "ns not found" }
```

то необходимо создать новую коллекцию.

- создание новой коллекции с лимитом по объёму в байтах:

```
>db.createCollection("eventLog", {capped:true, size: 100000000})
```

- создание новой коллекции без лимита:

```
>db.createCollection("eventLog")
```

# Обязанности и задачи администратора

4. Повторно проверить создание коллекции командой:

```
>db.eventLog.stats()
```

Ответ должен выглядеть примерно следующим образом:

```
{  
  
    "ns" : "saymon.eventLog",  
  
    "count" : 0,  
  
    "size" : 0,  
  
    "storageSize" : 100003840,  
  
    "numExtents" : 1,  
  
    "nindexes" : 1,  
  
    "lastExtentSize" : 100003840,  
  
    "paddingFactor" : 1,  
  
    "systemFlags" : 1,  
  
    "userFlags" : 0,  
  
    "totalIndexSize" : 8176,  
  
    "indexSizes" : {  
  
        "_id_" : 8176  
  
    },  
  
    "capped" : true,  
  
    "max" : NumberLong("9223372036854775807"),  
  
    "ok" : 1  
  
}
```

## 5.2.7 Удаление логов

Удаление логов осуществляется следующей командой в консоли:

```
sudo rm -rf /var/log/upstart/*
```

# Обязанности и задачи администратора

или при помощи REST API:

```
/node/api/event-log/:id
```

## 5.3 Управление учётными записями пользователей

### 5.3.1 Создание учетных записей

Создание учётных записей пользователей осуществляется двумя способами:

- При помощи REST API, например:

```
{
  "login": "Bob",
  "password": "password",
  "permissions": [
    "manage-objects",
    "manage-links"
  ]
}
```

- Посредством Web-UI:

- В панели инструментов нажать на имя пользователя и в выпадающем меню выбрать пункт "Конфигурация".

- Добавить пользователя или группу нажатием соответствующей кнопки 

- Выбрать из выпадающего списка объект добавления.

- При добавлении пользователя необходимо заполнить поля:

"Логин", "Пароль" и его подтверждение. Нажать на кнопку ("Добавить").



### 5.3.2 Назначение пользователям прав доступа

В зависимости от цели использования "Центрального пульта" администратор системы имеет возможность добавлять/ограничивать права доступа пользователям.

Существует два способа изменения прав доступа:

- При помощи REST API, например:

```
{
  "permissions": [
```

# Обязанности и задачи администратора



## 2. Посредством Web-UI:

- 2.1. В панели инструментов нажать на имя пользователя и в выпадающем меню выбрать пункт "Конфигурация".
- 2.2. В списке пользователей и групп выбрать объект изменения настроек.
- 2.3. Перейти во вкладку "Права на операции".
- 2.4. Если необходимо, разрешить/запретить все права и просмотр всех секций нажатием соответствующих кнопок.
- 2.4. Выборочно добавить/ограничить права и просмотр секций при помощи слайдера.

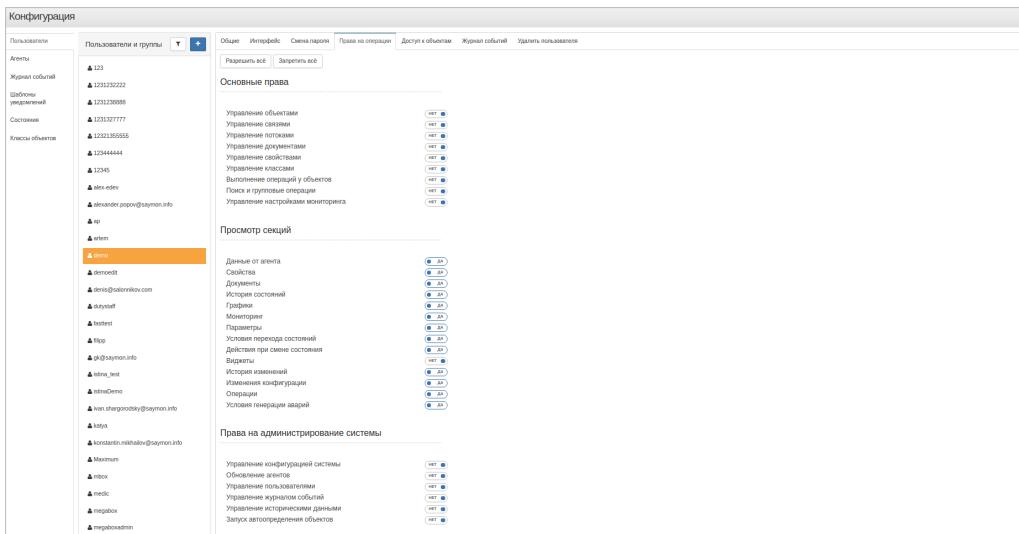


Рисунок.8. Права пользователей

### 5.3.3 Изменение пароля от учётной записи

Пользователь может менять пароль от своей учётной записи:

1. Самостоятельно через Web UI:
  - 1.1. В панели инструментов нажать на имя пользователя и в выпадающем меню выбрать пункт "Конфигурация".
  - 1.2. В списке пользователей и групп выбрать объект изменения настроек.

# Обязанности и задачи администратора

1.3. Ввести текущий пароль, новый пароль и подтвердить его в соответствующих строках.

1.4. Сохранить изменения.

2. Путём обращения к администратору и использования REST API:

```
{
  "currentPassword": "password",
  "newPassword": "newPassword"
}
```

## 5.3.4 Удаление пользователя

1. REST API метод:

```
/node/api/users/:id
```

2. Через Web UI:

2.1. В панели инструментов нажать на имя пользователя и в выпадающем меню выбрать пункт "Конфигурация".

2.2. В списке пользователей и групп выбрать объект изменения настроек.

2.3. Удалить пользователя нажатием соответствующей кнопки.

2.4. Во всплывающем окне подтвердить или отменить удаление.

## 5.4 Работа с объектами и связями

### 5.4.1 Создание объекта

Создание объекта, в пределах которого осуществляется мониторинг с целью сбора информации, её анализа и контроля за состоянием объекта, осуществимо тремя способами:

1. Из терминала следующей командой:

```
curl -H "Content-Type: application/json" -X POST -d '{"name": "My new object",
"parent_id": "1", "clas
```

где:

Параметр	Описание
"name": "My new object"	Имя создаваемого объекта, например, My new object.
"parent_id": "1"	ID объекта, в котором будет создан новый объект, например, 1.

# Обязанности и задачи администратора

Параметр	Описание
"class_id": "3"	ID класса создаваемого объекта, например, 3.
-u login:password	Имя и пароль пользователя, например, login и password.
https://saas.saymon.info/node/api/objects	Адрес сервера системы, например, https://saas.saymon.info, и соответствующий метод REST-API - /node/api/objects

2. С помощью REST API, например:

```
{
  "name": "My new object",
  "parent_id": 1,
  "class_id": 3
}
```

3. Через Web UI:

3.1. Приступить к созданию объекта нажатием кнопки

**+ Создать объект** на панели инструментов.

3.2. В появившемся всплывающем окне «Новый объект» следует заполнить поле «Имя объекта» и выбрать класс объекта.

Рисунок.9. Создание нового объекта

3.3. Подтвердить создание нажатием соответствующей кнопки

**Создать**

# Обязанности и задачи администратора

## 5.4.2 Клонирование объекта

Выбранный объект клонируется со всеми свойствами, документацией, связями и дочерними объектами.

Клонирование реализуется следующим образом:

1. С использованием REST API метода:

```
/node/api/objects/:id/clone
```

2. Через Web UI:

- 2.1. Правой кнопкой мыши нажать на объект клонирования.
- 2.2. В контекстном меню объекта выбрать действие "Клонировать".

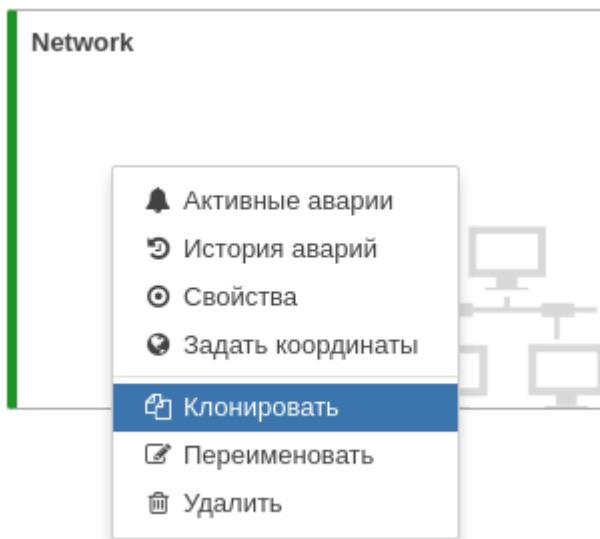


Рисунок.10. Клонирование объекта

## 5.4.3 Удаление объекта

Чтобы удалить объект, существует несколько способов:

1. REST API метод с указанием ID объекта:

```
/node/api/objects/:id
```

2. С помощью контекстного меню объекта в Web UI:

- 2.1. Правой кнопкой мыши нажать на объект клонирования.
- 2.2. В контекстном меню объекта выбрать действие "Удалить".

# Обязанности и задачи администратора

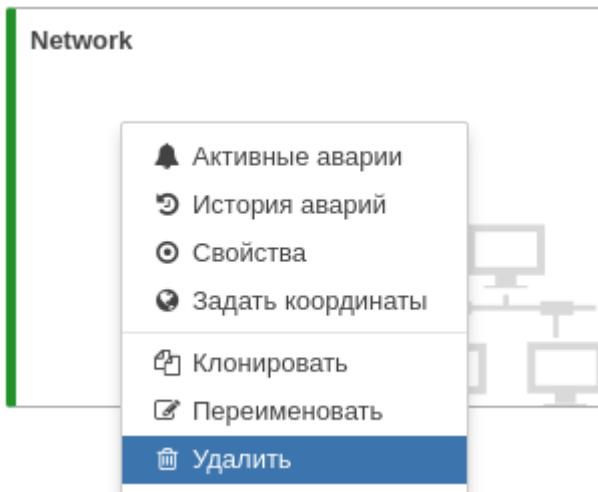


Рисунок.11. Удаление объекта

3. С использованием режима удаления в Web UI:

3.1. Активировать режим удаления нажатием соответствующей кнопки  на панели инструментов.

3.2. Нажать на такой же значок  на выбранном объекте для его удаления.

3.3. Выйти из режима удаления повторным нажатием на кнопку .

## 5.4.4 Создание ссылки на объект

Ссылка представляет собой особый тип объекта и служит для отображения уже настроенных в инфраструктуре объектов в других её частях, например, дашбордах.

Создать ссылку возможно двумя способами:

1. Путём REST API метода:

```
/node/api/refs
```

2. Через Web UI:

- 2.1. Нажать на кнопку  на панели инструментов.
- 2.2. В выпадающем списке появившегося окна выбрать объект, для которого создаётся ссылка.

# Обязанности и задачи администратора

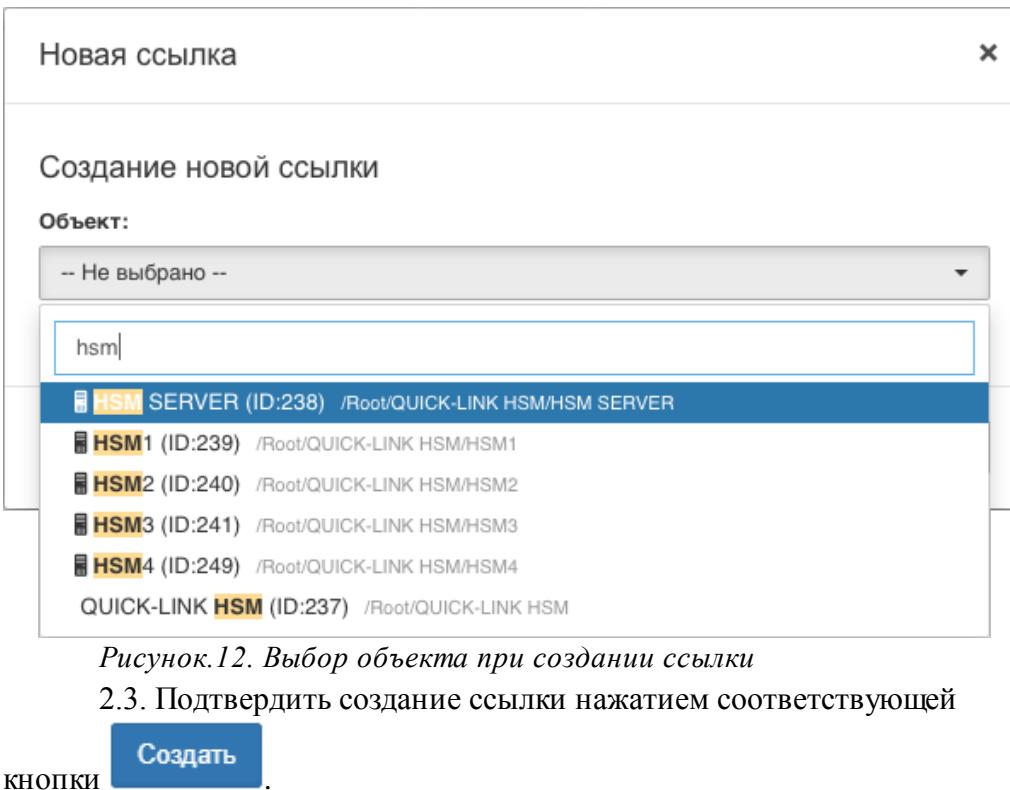


Рисунок.12. Выбор объекта при создании ссылки

2.3. Подтвердить создание ссылки нажатием соответствующей

кнопки

**Создать**

## 5.4.5 Создание связи

Связи между объектами также могут являться объектами мониторинга. Для их создания можно воспользоваться одним из методов:

1. REST API метод:

```
/node/api/links
```

2. Создание связи через Web UI:

2.1. Активировать режим создания связи соответствующей

кнопкой  на панели инструментов.

2.2. После того, как на всех объектах появится соответствующий символ, нажать на него на исходном объекте и, удерживая, переместить курсор на целевой объект.



# Обязанности и задачи администратора

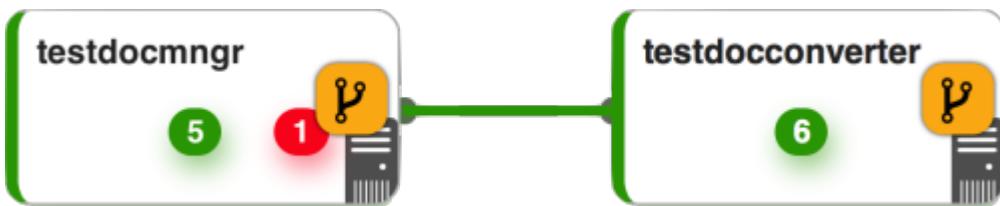


Рисунок.13. Создание связи

2.3. Отключить режим создания связи повторным нажатием

кнопки

## 5.4.6 Удаление связи

Данная операция реализуема двумя способами:

1. Через REST API метод с указанием ID связи:

`/node/api/links/:id`

2. В Web UI:

2.1. Активировать режим удаления нажатием соответствующей

кнопки на панели инструментов.

2.2. Нажать на такой же значок на выбранной связи для её удаления.

2.3. Выйти из режима удаления повторным нажатием на кнопку



## 5.5 Настройка уведомлений

В "Центральном пульте" реализованы различные способы уведомлений:

- отправлять email-уведомления;
- автоматически запускать программу или скрипт с параметрами;
- отправлять сообщения в Telegram;
- отправлять SMS;
- совершать голосовые вызовы;
- показывать визуальное уведомление в браузере, сопровождающееся звуком.

Уведомления устанавливаются в веб-интерфейсе в *Параметрах* объекта во вкладке *Действия при смене состояния*.

# Обязанности и задачи администратора

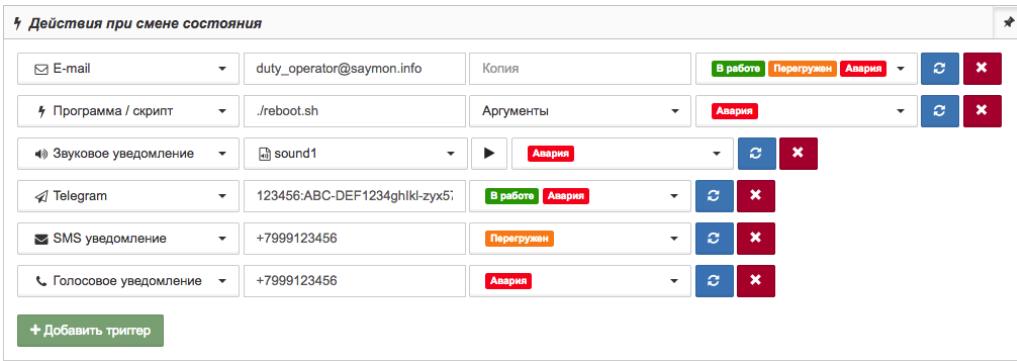


Рисунок.14. Действия при смене состояний

## 5.5.1 Активация функционала отправки SMS и голосовых вызовов

Для активации функционала отправки SMS-уведомлений и голосовых вызовов необходимо выполнить описанные ниже действия:

1. Создать копию файла настроек по умолчанию:

```
cp /usr/local/saymon/target/client/js/default-client-
config.js /usr/local/saymon/target/client/js/client-config.js
```

2. В новом файле индивидуальных настроек

- */usr/local/saymon/target/client/js/client-config.js* - добавить параметры **enableSmsTrigger: true** и **enableVoiceCallTrigger: true** следующим образом:

```
define(['utils/common'], function(common) {
    return {
        enableVoiceCallTrigger: true,
        enableSmsTrigger: true,
        ...
    }
});
```

3. В конфигурационном файле сервера системы - */etc/saymon/saymon-server.conf* - в разделе *server* задать путь до скриптов, осуществляющих отправку SMS-сообщений и головой вызов:

```
"sms_script": "путь до скрипта, отправляющего sms-уведомления"
"voice_call_script": "путь до скрипта, осуществляющего голосовой вызов"
```

4. Обновить вкладку для применения изменений.

## 5.5.2 Отправка почтовых уведомлений

Для настройки отправки почтовых уведомлений требуется добавить необходимые параметры доступа к почтовому серверу в конфигурационный файл - */etc/saymon/saymon-server.conf* - в подраздел *email*:

```
"disables" : "отправка почтовых уведомлений (false или true, по умолчанию true)"
```

# Обязанности и задачи администратора

```

"transport" {}
  "host" : "адрес почтового сервера, например, smtp.gmail.com",
  "port" : "порт, например, 465",
  "secure" : "true или false",
  "auth" {}
    "user" : "логин пользователя для доступа к почтовому серверу"
    "pass" : "пароль пользователя для доступа к почтовому серверу"
  "fields" {}
    "from" : "почтовый адрес отправителя, например,
saymon@saas.saymon.info"
    "max_json_length": "ограничение размера письма с уведомлением в
символах (если не указано - 1000 по умолчанию)"

```

## 5.5.3 Настройка уведомлений в Telegram

Для настройки отправки уведомлений с помощью Telegram необходимо:

1. Создать бота, от которого будут приходить уведомления:

- найти в приложении Telegram контакт **BotFather**;
- отправить ему сообщение **/newbot**;
- задать ему отображаемое имя (**name** - можно позже поменять);
- задать ему уникальное имя (**username** - поменять невозможно);
- скопировать токен бота вида:

**210979209:AAFft2mt3oW4EK1gYqE\_d3OjAJSIRLSrALo.**

2. Создать канал, чат или начать работу с группой:

2.1. Приватный канал (рекомендуется).

- в приложении Telegram создать новый канал;
- в процессе создания скопировать ID канала вида **AAAAAEJ-fsRCxgBXqUCgCq**;
- открыть настройки канала и добавить бота в список администраторов.

2.2. Создать чат.

- отправить боту любое сообщение;
- перейти по ссылке [api.telegram.org/bot<токен\\_бота\\_смотри\\_выше>/getUpdates](https://api.telegram.org/bot<токен_бота_смотри_выше>/getUpdates) в любом веб-браузере, вставив в неё токен своего бота без пробелов и знаков <>;
- найти текст со словами "chat" и "id", например, ... **"K","chat": {"id":121399918,"first\_...**;
- цифры **121399918** - необходимый ID чата.

2.3. Работа с группой.

# Обязанности и задачи администратора

- добавить бота в группу;
- отправить боту в группу любое сообщение, начав его со знака @;
- перейти по ссылке  
`api.telegram.org/bot<токен_бота_смотри_выше>/getUpdates` в любом веб-браузере, вставив в неё токен своего бота без пробелов и знаков <>;
- найти текст со словами chat и id, например, ... ":"K"},"chat":{"id":-209194473,"first\_...;
- цифры -209194473 - требуемый ID группы.

3. Вернуться к настройкам уведомлений в веб-интерфейсе системы в *Параметрах объекта* во вкладке *Действия при смене состояний* и выбрать Telegram в выпадающем списке.

4. Указать токен бота и ID канала (или ID чата/группы) в соответствующих полях.

## 5.6 Настройка интерфейса

### 5.6.1 Выравнивание / расстановка объектов в Стандартном виде

Выравнивание объектов в Стандартном виде реализуется за счёт сетки. Чтобы настроить шаг и цвет сетки, необходимо:

1. Создать копию файла настроек по умолчанию:

```
cp /usr/local/saymon/target/client/js/default-client-
config.js /usr/local/saymon/target/client/js/client-config.js
```

2. В новом файле индивидуальных настроек в секции *grid* указать требуемые шаг и цвет сетки, например:

```
grid: {
  dim: 20, // размер
  color: 'rgb(128, 128, 128)' // цвет
}
```

3. Обновить страницу браузера.

Включение сетки осуществляется в панели "Хлебные крошки" нажатием соответствующей кнопки . После чего объекты расставляются исключительно по отображаемой сетке.

# Обязанности и задачи администратора

## 5.6.2 Отображение названия компании в заголовке веб-интерфейса

При желании в заголовке веб-интерфейса системы возможно отобразить название компании. Для этого требуется:

1. Создать копию файла настроек по умолчанию:

```
cp /usr/local/saymon/target/client/js/default-client-
config.js /usr/local/saymon/target/client/js/client-config.js
```

2. В новом файле индивидуальных настроек

- */usr/local/saymon/target/client/js/client-config.js* - указать требуемое название в строке:

```
title: '<your-new-name>'
```

3. Обновить страницу браузера.

## 5.6.3 Перемещение и отключение фоновой иконки у объекта

Каждый класс объекта имеет индивидуальную фоновую иконку. При создании объекта и выборе класса, она автоматически отображается на созданном объекте. Если она доставляет неудобства при работе, то для её перемещения необходимо в нижнюю секцию на вкладке *Стили в Параметрах* объекта добавить:

```
.object-background {
background-position: 9px 8px !important;
background-size: 15px;
opacity: 1;
}

.object-caption-panel {
padding-left: 15px;
}
```

Для отключения иконки требуется добавить в ту же секцию:

```
.background-component {
display: none
}
```

# Обязанности и задачи администратора

## 5.6.4 Вертикальное отображение названия объекта

Как правило, объекты подписываются в правом левом углу. Для вертикального отображения названия объекта требуется в *Параметрах* объекта на вкладке *Стили* добавить в нижнюю секцию после линии код:

```
.js-caption {
writing-mode: vertical-lr;text-orientation: upright;text-transform: uppercase;letter-spacing: 1px;
}
```

## 5.6.5 Редактирование стилей состояний

В процессе мониторинга в зависимости от данных, получаемых от агента, объект может менять состояние. По умолчанию система оперирует следующими состояниями:

- Авария на объекте
- Объект перегружен
- Нет данных по объекту
- Объект не функционирует
- Объект в работе
- Объект создан
- Объект не проверяется

С целью визуализации каждый объект имеет свой цвет. Цвет и стиль состояния допускается видоизменять следующими способами:

1. Путём изменения конфигурационного файла:

1.1. Создать файл:

/usr/local/saymon/saymon.local/css/saymon.local.css

1.2. Открыть его в текстовом редакторе и вставить код для изменения цвета объекта и его фона, например:

```
.state-4 {
background-color: rgba(255, 122, 0, 0.46);
box-shadow: 2px 5px 10px rgba(253, 118, 7, 0.5);
border-left: 5px solid #FD7607;
}

.state-bg-4 {
background-color: #FD7607;
}

.view-screen-element ul li.state-4 {
border-left: 5px solid #FD7607;
```

# Обязанности и задачи администратора

}

```
.view-screen-element ul li .badge.state-4 {
background-color: #FD7607;
```

Номер состояния соответствует его ID.

2. Через Web UI:

- 2.1. В панели инструментов нажать на имя пользователя и в выпадающем меню выбрать пункт "Конфигурация".
- 2.2. Перейти в раздел "Состояния".
- 2.3. В списке состояний выбрать то, которое требуется изменить.
- 2.4. При необходимости изменить имя в соответствующем поле.
- 2.5. Настроить основной цвет, цвет тени, цвет строки таблицы и фон объекта при помощи цветовой палитры или методом ввода номера цвета.

## 5.7 Настройка мониторинга

"Центральный пульт" позволяет осуществлять процесс мониторинга с использованием различных типов проверок. Проверки настраиваются в веб-интерфейсе системы. Для получения информации о каждом типе рекомендуется ознакомиться с Руководством пользователя "Центрального пульта".

Ниже рассмотрены несколько примеров процесса мониторинга.

### 5.7.1 Мониторинг основных параметров ПК

Для мониторинга основных параметров работы сервера или ПК: CPU, File System, Memory и Network IO, достаточно выполнить несколько действий:

1. Установить агента на наблюдаемый ПК или сервер;
2. Создать объект, например, класса *Host* в веб-интерфейсе;
3. Перейти в созданный объект *Host* и добавить внутри него объекты классов: *SAYMON Agent*, *CPU*, *File System*, *Memory* и *Network IO*.
4. Сконфигурировать и запустить агента.

Через некоторое время информация об основных параметрах работы компьютера начнёт поступать на сервер и отображаться в веб-интерфейсе системы.

# Обязанности и задачи администратора

## 5.7.2 Мониторинг процесса памяти

Для настройки мониторинга процесса в памяти сервера или ПК необходимо:

1. Установить, сконфигурировать и запустить агента на наблюдаемом ПК или сервере;
2. Создать объект, например, класса *Process* в веб-интерфейса и перейти к его настройкам;
3. В секции Мониторинг требуется:
  - выбрать агента, установленного ранее на данный компьютер;
  - выбрать тип проверки *Процесс по имени*;
  - заполнить необходимые поля.

Через некоторое время информация о проверяемых процессах начнёт поступать на сервер и отображаться в веб-интерфейсе системы.

## 5.7.3 Мониторинг изменения файлов и папок

Для настройки мониторинга изменения файлов и папок сервера или ПК необходимо:

1. Установить, сконфигурировать и запустить агента на наблюдаемом ПК или сервере;
2. Создать объект класса *Configuration File* в веб-интерфейсе и перейти к его настройкам;
3. В секции Мониторинг:
  - выбрать агента, установленного ранее на данный компьютер;
  - выбрать тип проверки *Конфигурационный файл / директория*;
  - указать путь к проверяемой папке или файлу.

Через некоторой время информация о проверяемых файлах начнёт поступать на сервер и отображаться в веб-интерфейсе системы.

## 5.7.4 Проверка доступности веб-ресурса

Данный тип мониторинга позволяет убедиться не только в работоспособности веб-сайта (статус 200 OK), но и в ограничении доступа к таким ресурсам, как панель администрирования баз данных. В этом случае статус 403 Forbidden или 404 Not Found будет говорить о правильности настройки системы, а иной статус - о возможной угрозе безопасности системы.

Для проверки доступности и скорости отклика веб-ресурса необходимо:

# Обязанности и задачи администратора

1. Установить, сконфигурировать и запустить хотя бы одного агента в инфраструктуре;
2. Создать объект, например, класса *Address* в веб-интерфейса и перейти к его настройкам;
3. В секции Мониторинг требуется:
  - выбрать агента, который будет выполнять проверку;
  - выбрать тип проверки *HTTP-запрос*;
  - выбрать тип запроса *GET*;
  - указать адрес веб-сайта в поле URL.

Через некоторое время информация о доступности и скорости отклика наблюдаемого веб-ресурса начнёт поступать на сервер и отображаться в интерфейсе системы.

## 5.7.5 Безагентный мониторинг веб-сервера

Существует ряд случаев, при которых установка агента на сервере невозможна. В таких случаях рекомендовано написать скрипт, который с заданной периодичностью будет выполняться на сервере, собираясь необходимые данные и генерировать текстовый файл с результатами в JSON-формате по ссылке, доступной извне.

Для мониторинга параметров веб-сервера, на который невозможно поставить агента, необходимо:

1. Написать локальный скрипт, выполняющий подготовку данных (например, в папке загрузок: `../downloads/scripts/webserver_stat.sh`):

```
#!/bin/bash
# Сбор параметров работы веб-сервера.

# использование Memory
memUsage=$(free -m | grep Mem | perl -pe 's/Mem:\s+\S+\s+(\S+).*/$1/')

# использование Swap
swapUsage=$(free -m | grep Swap | perl -pe 's/Swap:\s+\S+\s+(\S+).*/$1/')

# загрузка CPU
cpuUsage=$(uptime | awk '{print $10}' | perl -pe 's/,//')

# проверка выполнения какого-либо скрипта, например, webserver_stat.sh
scriptExec=$(ps -ef | grep webserver_stat.sh | grep -v grep | wc -l)

# Write JSON response
```

# Обязанности и задачи администратора

```
echo "{\"memUsageMB\": \"$memUsage\", \"swapUsageMB\": \"$swapUsage\",
\"cpuUsage\": \"$cpuUsage\", \"scriptExec\": \"$scriptExec\"} " > webserver_stat.json
```

2. Добавить выполнение скрипта в планировщик заданий cron;
3. Установить, сконфигурировать и запустить хотя бы одного агента, который будет осуществлять сбор данных;
4. Создать объект, например, класса *Info* в веб-интерфейсе и перейти к его настройкам;
5. В секции Мониторинг:
  - выбрать агента, который будет выполнять проверку;
  - выбрать тип проверки *HTTP-запрос*;
  - выбрать *GET* в поле *Тип запроса*;
  - в поле *URL* ввести адрес JSON-файла  
[http://saymon.info/downloads/scripts/webserver\\_stat.json](http://saymon.info/downloads/scripts/webserver_stat.json).

Через некоторое время информация о параметрах работы веб-сервера начнёт поступать на сервер и отображаться в интерфейсе системы.

## 5.8 Ипорт и экспорт данных

В системе предусмотрен перенос основных данных между различными инсталляциями.

Для экспорта данных из MongoDB и MySQL в специальный архив на существующей инсталляции системы необходимо запустить скрипт:

```
/usr/local/saymon/backend/scripts/migration/export.sh
```

Для импорта данных из MongoDB и MySQL на новой инсталляции системы необходимо подать данный архив на вход скрипту:

```
/usr/local/saymon/backend/scripts/migration/import.sh
```

# **Проблемы в работе системы и способы их решения**

**ЦЕНТРАЛЬНЫЙ ПУЛЬТ**

## 6 Проблемы в работе системы и способы их решения

### 6.1 Недостаточно места на виртуальной машине с сервером

#### Условия проблемы:

- Есть виртуальная машина с SAYMON Server
- Нет места на виртуальной машине с SAYMON Server

#### Решение проблемы:

1. Понять содержимое и объём занимаемого места:

```
sudo du -h / | sort -h
```

2. Просмотреть список папок в *stdout*:

- если много места занимает папка */var/log/saymon*, то можно уменьшить количество хранимых лог-файлов правкой */etc/logrotate.d/saymon* для *saymon-server.log: rotate X* и */opt/saymon-agent/conf/logback-upstart.xml* для *saymon-agent.\*.log:<maxHistory>10</maxHistory>*;
- если много места занимают данные из MongoDB, то зайти в базу данных и оценить размеры коллекций:

```
mongo saymon
```

```
function getReadableFileSizeString(fileSizeInBytes) {
    var i = -1;
    var byteUnits = [' kB', ' MB', ' GB', ' TB', ' PB', ' EB', ' ZB', ' YB'];
    do {
        fileSizeInBytes = fileSizeInBytes / 1024;
        i++;
    } while (fileSizeInBytes > 1024);
    return Math.max(fileSizeInBytes, 0.1).toFixed(1) + byteUnits[i];
}
var collectionNames = db.getCollectionNames(), stats = [];
collectionNames.forEach(function (n) { stats.push(db.getCollection(n).stats()); });
stats = stats.sort(function(a, b) { return b['size'] - a['size']; });
for (var c in stats) { print(stats[c]['ns'] + ":" + getReadableFileSizeString(stats[c]['size']) + "(" + getReadableFileSizeString(stats[c]['storageSize']) + ")"); }
```

В наиболее объёмных коллекциях используется timestamp, следующей командой можно удалить из коллекции stateHistory массив данных за рамками глубины хранения:

```
db.stateHistory.remove({timestamp:{$gt:1477994233000}})
```

После выше описанных действий место в системе не освободится, так как MongoDB аллоцирует дисковое пространство. Требуется сделать бекап и восстановить базу:

```
mongodump
sudo rm -rf /var/lib/mongodb/*
sudo mongorestore dump/ --dbpath /var/lib/mongodb/
sudo chown -R mongodb:mongodb /var/lib/mongodb
sudo service mongod restart
```

- если много места занимают данные Open TSDB, не вынесенные из Docker-контейнера. Их можно вынести:

```
sudo docker exec -it opentsdb bash
cd /data/hbase/hbase-root
tar zcvf hbase-root.tar.gz hbase-root
scp hbase-root.tar.gz saymon@*host_ip*:opt/.
exit
cd /opt/ && tar xvf hbase-root.tar.gz
sudo docker stop opentsdb
sudo docker rm opentsdb
sudo docker run -d -p 127.0.0.1:4242:4242 --restart=always --
volume /opt/hbase-root:/data/hbase/hbase-root/ --name=opentsdb
rossinno/opentsdb
```

## 6.2 Отсутствие подключения агента к серверу

### Условия проблемы:

- агент не подключается к серверу;
- запись в логе: 12.10.2016 07:45:59.431 [pool-1-thread-1] WARN n.r.s.agent.connection.RedisBackend - Redis connection failed (will retry in 5 seconds): JedisDataException: ERR max number of clients reached

### Решение проблемы:

1. Проверить на сервере проблему локально:

```
# redis-cli -a 'пароль_от_redis_в_кавычках' info clients | grep connected_clients
| sed -e 's/connected_clients//g'

Error: Connection reset by peer
```

2. Проверить проблему локально через redis-cli:

```
# redis-cli
```

```
127.0.0.1:6379> auth пароль_от_redis
(error) ERR max number of clients reached
127.0.0.1:6379> q
```

3. Рестарт Redis-сервера:

```
# service redis-server restart
Stopping redis-server: redis-server.
Starting redis-server: redis-server
```

## 6.3 Проверка работы MongoDB

**Проверка наличия процесса в памяти:**

```
ps -ef | grep mongod
mongodb 1147 1 0 Nov02 ? 04:23:16 /usr/bin/mongod --
config /etc/mongod.conf
```

**Остановка, запуск и рестарт процесса:**

```
sudo service mongod status
sudo service mongod start / stop
sudo service mongod restart
```

## 6.4 Проверка работы MySQL

**Проверка пароля MySQL (действие на хосте с сервером):**

```
cat /etc/saymon/saymon-server.conf
```

**Просмотр секции db{ }:**

```
"db" : {
  "host" : "localhost",
  "user" : "user",
  "password" : "password",
  "database" : "saymondb"
},
```

## 6.5 Проверка работы Redis

**Проверка наличия процесса в памяти:**

```
ps -ef | grep redis
```

```
redis 1763 1 0 Aug10 ? 00:37:11 /usr/bin/redis-server 0.0.0.0:6379
root 1786 1 0 Aug10 ? 00:00:00 /usr/bin/stunnel4 /etc/stunnel/redis-client.conf
root 1787 1 0 Aug10 ? 00:00:00 /usr/bin/stunnel4 /etc/stunnel/redis-client.conf
...
```

**Остановка, запуск и рестарт процесса:**

```
sudo service redis-server stop/start/restart
```

**Номер порта, на котором осуществляется процесс:**

```
sudo netstat -lnp | grep redis
```

```
tcp 0 0 0.0.0.0:6379 0.0.0.0:* LISTEN 1763/redis-server 0
```

или в конфигурационном файле:

```
cat /etc/saymon/saymon-server.conf | grep cache -A 4
```

```
"cache": {
    "auth_pass": "12!@easy",
    "host": "127.0.0.1",
    "port": 6379
},
```

**Проверка доступности (открытости) порта:**

```
sudo iptables -L INPUT -n -v --line-numbers
```

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num pkts bytes target prot opt in out source destination
1 15M 3082M ACCEPT tcp - * * 0.0.0.0/0 0.0.0.0/0 tcp dpt:6379
```

**Добавление порта в список открытых и запись нового правила:**

```
sudo iptables -I INPUT 1 -m state --state NEW -p tcp --dport 6379 -j ACCEPT
```

```
sudo bash -c "/sbin/iptables-save > /etc/iptables.rules"
```

**Проверка доступности порта для агента (действие на хосте с агентом):**

```
telnet <адрес_сервера> 6379
```

```
Trying <адрес_сервера>...
```

Connected to <адрес\_сервера>. Escape character is '^]'.

#### **Проверка пароля Redis (действие на хосте с сервером)**

- На хосте с сервером (конфигурация Redis):

```
cat /etc/redis/redis.conf | grep requirepass
requirepass Ja!MIK1&
# If the master is password protected (using the "requirepass" configuration
# requirepass foobared
```

- На хосте с сервером (конфигурация SAYMON):

```
cat /etc/saymon/saymon-server.conf | grep auth_pass
"auth_pass" : "Ja!MIK1&"
```

- На хосте с агентом:

```
cat /opt/saymon-agent/conf/agent.properties | grep password
server.password=Ja!MIK1&
```

Пароли должны совпадать, иначе агент не сможет подключиться к серверу для отправки данных.

## **6.6 500 Internal server error и отсутствие графиков**

#### **Условие проблемы:**

- вместо графиков возникает ошибка 500

#### **Решение проблемы:**

Необходимо перезапустить OpenTSDB

- less /var/log/opentsdb/opentsdb.log (тут можно увидеть какие-то ошибки)
- sudo service opentsdb stop
- sudo service hbase restart
- sudo service opentsdb start

## **6.7 Ошибка работы HTTP-проверки**

#### **Условие работы:**

- HTTP-проверка адреса https://xxx.xxx не работает и возникает ошибка

## **Решение проблемы:**

Данная проблема возникает при использовании агента в связке с Java 1.6.

Существует 2 варианта решения:

1. Обновить Java, установленную в операционной системе, до версии 1.7 или 1.8.
2. Скачать и установить последнюю версию агента со встроенной Java.

**РОССИННО**

**[www.rossinno.net](http://www.rossinno.net)**

**2019**