



# Руководство администратора платформы "Центральный Пульт"

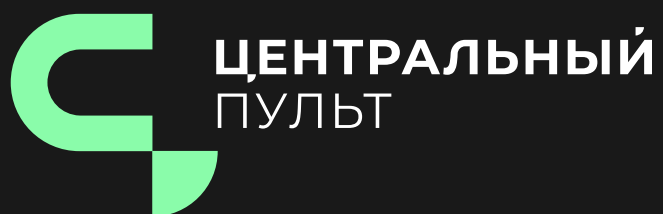
v.3.7.81 - 01.04.2021



# Оглавление

1. Описание документа .....	4
2. Назначение системы .....	6
3. Системные требования .....	8
3.1. Системные требования к серверу .....	9
3.2. Системные требования к агенту .....	10
4. Принципы функционирования системы .....	12
4.1. Архитектура системы .....	12
4.2. Описание функционирования системы и её частей .....	15
5. Задачи администратора .....	17
5.1. Настройка параметров работы системы .....	18
5.1.1. Установка агента .....	19
5.1.1.1. Автоинсталляция .....	20
5.1.1.2. Linux .....	21
5.1.1.3. Mac OS X .....	22
5.1.1.4. Windows .....	23
5.1.1.5. Wirenboard 6 .....	24
5.1.2. Конфигурация агента .....	25
5.1.3. Конфигурация сервера .....	27
5.1.4. Опции конфигурации .....	37
5.1.5. Служба "saymon-server" .....	39
5.1.6. Просмотр websocket-нотификаций .....	40
5.1.7. Увеличение количества обработчиков SNMP-Trap .....	41
5.1.8. Сброс системы к заводским настройкам .....	42
5.2. Управление логированием .....	43
5.2.1. Конфигурация log-файлов .....	44
5.2.2. Конфигурация ротации log-файлов .....	45
5.2.3. Просмотр информации о Журнале событий .....	46
5.2.4. Назначение ответственного за событие .....	47
5.2.5. Установка ограничения для логирования .....	48
5.2.6. Удаление всех SNMP-Trap'ов из Журнала событий .....	49
5.2.7. Удаление логов .....	50
5.3. Управление учётными записями пользователей .....	51
5.3.1. Создание учётных записей .....	52
5.3.2. Назначение пользователям прав доступа .....	54
5.3.3. Изменение пароля от учётной записи .....	56

5.3.4. Удаление пользователя .....	57
5.4. Работа с объектами и связями .....	58
5.4.1. Создание объекта .....	59
5.4.2. Клонирование объекта .....	60
5.4.3. Удаление объекта .....	61
5.4.4. Создание ссылки на объект .....	62
5.4.5. Создание связи .....	63
5.4.6. Удаление связи .....	64
5.5. Настройка уведомлений .....	65
5.5.1. Активация функционала отправки SMS и голосовых вызовов .....	66
5.5.2. Отправка почтовых уведомлений .....	67
5.5.3. Настройка уведомлений в Telegram .....	68
5.6. Настройка интерфейса .....	70
5.6.1. Выравнивание/расстановка объектов в стандартном виде .....	71
5.6.2. Настройка заголовка web-интерфейса .....	72
5.6.3. Перемещение/отключение фоновой иконки объекта .....	73
5.6.4. Вертикальное отображение имени объекта .....	74
5.6.5. Редактирование стилей состояний .....	75
5.7. Настройка мониторинга .....	76
5.7.1. Мониторинг основных параметров ПК .....	77
5.7.2. Мониторинг процесса памяти .....	78
5.7.3. Мониторинг изменения файлов и папок .....	79
5.7.4. Проверка доступности web-ресурса .....	80
5.7.5. Безагентный мониторинг web-сервера .....	81
5.8. Импорт и экспорт данных .....	82
6. Проблемы в работе системы и способы их решения .....	84
6.1. Недостаточно места на виртуальной машине с сервером .....	85
6.2. Отсутствие подключения агента к серверу .....	87
6.3. Проверка работы MongoDB .....	88
6.4. Проверка работы MySQL .....	89
6.5. Проверка работы Redis .....	90
6.6. 500 Internal Server Error и отсутствие графиков .....	92
6.7. Ошибка работы HTTP-проверки .....	93

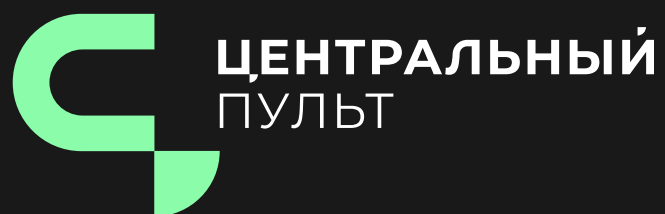


## **Описание документа**

## 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".



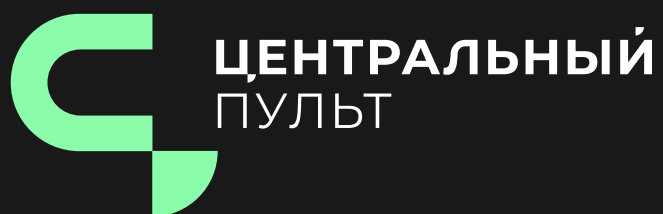
## **Назначение системы**

## 2 Назначение системы

Система предназначена для визуализации и мониторинга различных объектов. Центральный Пульс нацелен на упрощение сбора данных, ускорение их анализа, визуализации результатов и непрерывного 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**



## **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/MQTTS.

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: <https://disk.yandex.ru/d/EKWcd89n3NsADX>.

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

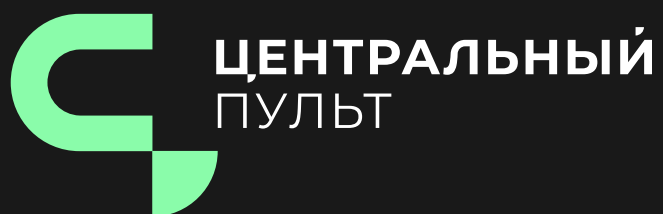
Агенты обладают кросс-платформенной совместимостью и могут быть установлены на различные операционные системы:

- 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**

## 4 Принципы функционирования системы

### 4.1 Архитектура системы

Платформа имеет клиент-серверную архитектуру и включает в себя три основных уровня (Рис. 1):

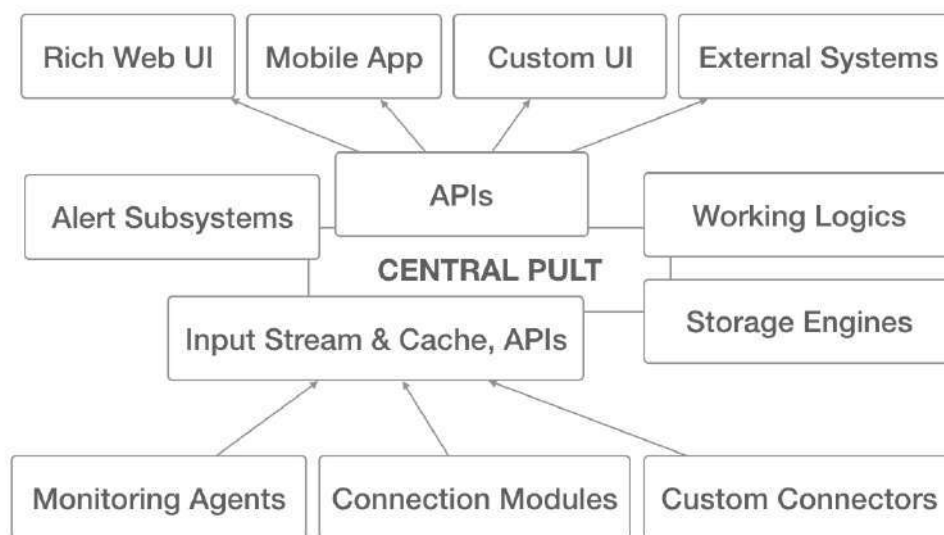


Рис. 1. Общая архитектура платформы "Центральный Пульс"

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:

- **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 Пульс может:
  - отправлять email-уведомления,
  - отправлять SMS,
  - отправлять сообщения в Telegram,
  - совершать голосовые вызовы,
  - 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 учётной 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,
  - передаёт данные серверу по запросу.

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 - web-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. Web-интерфейс имеет уникальное отображение, согласно заявленным требованиям.
- External Systems - внешние системы для отображения или сбора данных, полученных средствами мониторинга платформы "Центральный Пульс".

## 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.

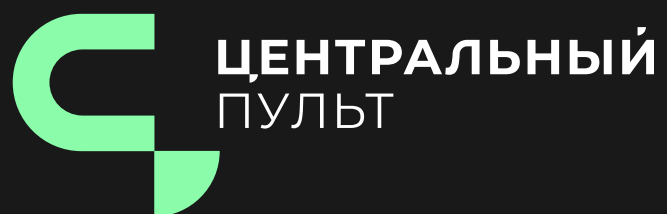
С помощью API в веб-интерфейсе системы возможно реализовать большинство операций, например:

- получение списка объектов;
- получение текущего статуса объектов и истории их состояний;
- запись данных в объекты без использования агентов;
- изменение свойств объектов;
- работа с инцидентами;
- 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 web -клиент системы и клиенты для мобильных платформ Android и iOS.





## **Задачи администратора**

## 5 Задачи администратора

Раздел содержит информацию об основных задачах, возникающих в процессе администрирования платформы.

## 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:

1. Получить актуальную версию агента одним из способов:

- зайти на сайт платформы в раздел "Загрузки" [saymon.info/support\\_ru/downloads\\_en/](http://saymon.info/support_ru/downloads_en/) и скачать подходящую под ОС сборку агента;
- 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.

2. Установить и настроить агента согласно дальнейшим инструкциям.

## 5.1.1.1 Автоинсталляция

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


1. Открыть web-интерфейс платформы "Центральный Пульт".
2. Ввести логин и пароль учётной записи с правами на управление объектами.
3. Создать объект-агент, для этого нажать кнопку **+ Создать объект** на панели инструментов, ввести имя объекта и выбрать класс "Saymon Agent".
4. Навести курсор на созданный объект и нажать на появившуюся иконку  - Настройки агента (Рис. 2):



Рис. 2. Мониторинговый агент

5. В появившемся окне скопировать ссылку из строки "Команда для установки агента" (Рис. 3):

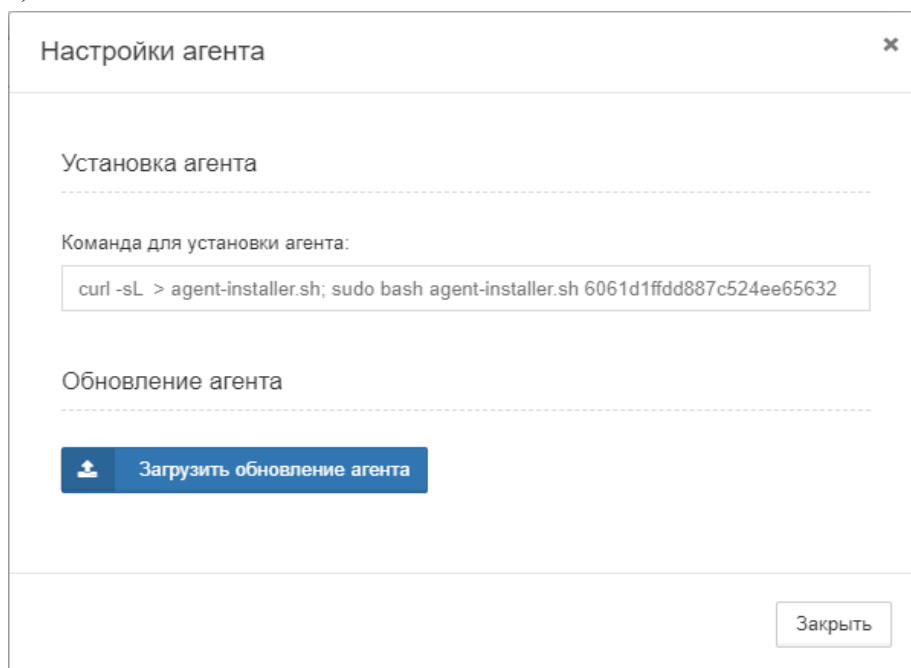
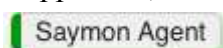


Рис. 3. Команда для установки агента

6. Выполнить эту команду в терминале на необходимом сервере.  
*Примечание: номер, который дал сервер - последние 24 символа команды, совпадает с ID агента.*

В течение 30 секунд агент скачивается и устанавливается. Если установка выполнена корректно, состояние агента в web-интерфейсе будет отражено зелёным цветом:



## 5.1.1.2 Linux

Инструкция применима к операционным системам с менеджерами служб "systemd", "upstart" и "init.d", например:

- Ubuntu Linux;
- Red Hat Enterprise Linux;
- CentOS Linux;
- Debian 8 "Jessie" (для пользователей Raspberry Pi).

Если вы используете другого менеджера, пишите нам на [care@saymon.info](mailto:care@saymon.info).

Для установки агента на Linux необходимо:

### На хосте без доступа в Интернет:

1. [Скачать sh-скрипт установщика агента](#) для Linux 64bit или Linux 32bit.
2. Перенести скрипт на нужный хост в домашнюю директорию пользователя.
3. Сделать скрипт исполняемым:

#### 64bit:

```
sudo chmod +x saymon-agent-rl-linux-x64-jre-installer.sh
```

#### 32bit:

```
sudo chmod +x saymon-agent-rl-linux-i586-jre-installer.sh
```

4. Запустить скрипт:

#### 64bit:

```
sudo ./saymon-agent-rl-linux-x64-jre-installer.sh
```

#### 32bit:

```
sudo ./saymon-agent-rl-linux-i586-jre-installer.sh
```

5. Выполнить дальнейшие инструкции в терминале.

### На хосте с доступом в Интернет:

1. Выполнить однострочник:

#### 64bit:

```
curl https://saymon.info/downloads/saymon-agent-rl-linux-x64-jre-installer.sh -o saymon-agent-instal.sh ; chmod +x saymon-agent-instal.sh ; sudo ./saymon-agent-instal.sh
```

#### 32bit:

```
curl https://saymon.info/downloads/saymon-agent-rl-linux-i586-jre-installer.sh -o saymon-agent-instal.sh ; chmod +x saymon-agent-instal.sh ; sudo ./saymon-agent-instal.sh
```

2. Выполнить дальнейшие инструкции в терминале.

Конфигурация агента выполняется в файле **"/opt/saymon-agent/conf/agent.properties"**.

## 5.1.13 Mac OS X

Для установки агента на Mac OS X необходимо:

1. Создать системного пользователя "Saymon agent", из-под которого будет запускаться агент.
2. [Скачать архив Mac OS X 64bit](#) и распаковать его в папку **"/opt"**.
3. Отредактировать файл конфигурации агента **"saymon-agent/conf/agent.properties"**.
4. Создать папку для хранения log-файлов:

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

5. При необходимости включить централизованное логирование агента через SYSLOG в файле **"saymon-agent/conf/logback.xml"**.
6. Сделать файл агента исполняемым:

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

7. Запустить агента:

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

## 5.1.14 Windows

Для установки агента на Windows необходимо:

1. [Скачать инсталлятор](#) для Windows 64bit или Windows 32bit.
2. Запустить скачанный файл от имени администратора.
3. Выбрать папку установки (например, "**C:\Program Files (x86)\SAYMON Agent**") и нажать кнопку "Install".
4. После завершения установки нажать кнопку "Close".
5. В папке установки агента из шага 3 отредактировать файл конфигурации агента "**...\conf\agent.properties**".
6. Запустить службу "SAYMON Agent".



### 5.1.15 Wirenboard 6

Wirenboard 6 - это универсальный контроллер для автоматизации с открытым ПО на базе Linux. Предназначен для домашней и промышленной автоматизации и мониторинга: опроса датчиков и счетчиков, использования в качестве УСПД, в системах АСКУЭ, для замены ПЛК, а также в системах "умного дома".

Для установки агента на контроллер автоматизации Wirenboard 6 необходимо:

1. [Скачать архив с агентом](#).
2. Распаковать архив в папку **"/opt/saymon-agent/"**.
3. [Скачать архив с JDK](#) (потребуется регистрации на сайте).
4. Распаковать архив в папку **"temp"**,  
оттуда полностью скопировать директорию **"jre"** в папку **"/opt/saymon-agent/"**.
5. Выполнить следующие команды:

```
useradd -M -r -s /bin/false -K MAIL_DIR=/dev/null saymon
chown -R saymon:saymon /opt/saymon-agent
cp /opt/saymon-agent/systemd/* /etc/systemd/system
systemctl enable saymon-agent
service saymon-agent start
```
6. При необходимости отредактировать файл конфигурации агента **"/opt/saymon-agent/conf/agent.properties"**.

Благодарим за оказанную помощь пользователя [svdu](#) с [форума компании Wiren Board](#).

### 5.1.2 Конфигурация агента

Конфигурация агента выполняется в файле  
`".../saymon-agent/conf/agent.properties"`.

Для применения изменений необходимо перезапустить службу "saymon-agent".

Далее приведены описания разделов и настроек.

Параметр	Описание
agent.discoveryEnabled	<p>Включает автоматический поиск агентом сетевых устройств;</p> <ul style="list-style-type: none"> <li>• false - автоматический поиск выключен,</li> <li>• true - автоматический поиск включен.</li> </ul> <p>По умолчанию - <b>false</b> - автоматический поиск выключен.</p> <p><i>Задать родительский объект для обнаруженных устройств можно с помощью параметра "discovery_parent_id" в разделе "Server" конфигурационного файла сервера.</i></p>
agent.id	<p>Уникальный идентификатор объекта класса "Saymon Agent" в web-интерфейсе системы.</p> <p><i>При указании agent.id=0 на сервере будет создан новый объект класса "Saymon Agent", и ID этого объекта будет передан агенту. На хосте с агентом будет создан файл ".../saymon-agent/storage/registration.json", где будет записан данный идентификатор.</i></p> <p><i>Чтобы заново инициировать процедуру получения ID, к примеру, если необходимо подключить агента к другому серверу, достаточно удалить файл "registration.json" и перезапустить агента.</i></p>
agent.optimizedThreadManagement	<p>Включает режим оптимизации использования потоков агентом;</p> <ul style="list-style-type: none"> <li>• false - режим оптимизации выключен,</li> <li>• true - режим оптимизации включен.</li> </ul> <p>По умолчанию - <b>false</b> - режим оптимизации выключен.</p>
agent.scriptsEnabled	<p>Включает выполнение агентом скриптов с указанным текстом;</p> <ul style="list-style-type: none"> <li>• true - выполнение скриптов включено.</li> <li>• true - режим оптимизации включен.</li> </ul> <p>По умолчанию - <b>true</b> - выполнение скриптов включено.</p>

Параметр	Описание
agent.snmpTrapEnabled	<p>Позволяет использовать агента в качестве получателя трапов;</p> <ul style="list-style-type: none"> <li>• false - получение трапов агентом выключено,</li> <li>• true - получение трапов агентом включено.</li> </ul> <p>По умолчанию - <b>false</b> - получение трапов выключено.</p>
agent.snmpTrapListenPort	<p>Порт для получения трапов.</p> <p>По умолчанию - <b>1162</b>.</p>
agent.snmpTrapReceiverThreadPoolSize	<p>Количество одновременных потоков для получения трапов.</p> <p>По умолчанию - <b>4</b>.</p>
server.host	<p>Адрес сервера системы.</p> <p>По умолчанию - <b>127.0.0.1</b>.</p>
server.password	<p>Пароль Redis.</p>
server.port	<p>Порт Kafka (9092) или Redis (6379), по которому осуществляется подключение агентов к серверу.</p>

### 5.13 Конфигурация сервера

Конфигурация сервера системы выполняется в файле  
"/etc/saymon/saymon-server.conf".

Для применения изменений необходимо перезапустить службу "saymon-server":

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

Далее приведены описания разделов и настроек.

Раздел/Параметр	Описание
Cache	В этом разделе задаются настройки сервера Redis.
cache.auth_pass	Пароль для доступа к серверу Redis.
cache.host	Адрес сервера Redis. По умолчанию - "127.0.0.1".
cache.port	Порт сервера Redis. По умолчанию - 6379.
Kafka	В этом разделе задаются настройки подключения к брокеру Kafka.
kafka.create_topics	Включает автоматическое создание топиков; <ul style="list-style-type: none"><li>• false - автоматическое создание выключено,</li><li>• true - автоматическое создание включено.</li></ul> По умолчанию - <b>false</b> - автоматическое создание выключено.
kafka.fetch_latest	Включает забор последних сообщений от брокера; <ul style="list-style-type: none"><li>• false - забор выключен,</li><li>• true - забор включен.</li></ul> По умолчанию - <b>false</b> - забор выключен.
kafka.host	Адрес брокера Kafka. По умолчанию - "localhost".

Раздел/Параметр	Описание
kafka.logLevel	<p>Уровень логирования Kafka;</p> <ul style="list-style-type: none"> <li>• 0 - NOTHING,</li> <li>• 1 - ERROR,</li> <li>• 2 - WARN,</li> <li>• 4 - INFO,</li> <li>• 5 - DEBUG.</li> </ul> <p>По умолчанию - <b>2</b> - WARN.</p>
kafka.port	<p>Порт брокера Kafka.</p> <p>По умолчанию - <b>9092</b>.</p>
kafka.requestTimeout	<p>Время ожидания ответа клиентом (в миллисекундах).</p> <p>По умолчанию - <b>30000</b> - 30 секунд.</p>
LDAP	<p>В этом разделе задаются параметры внешнего LDAP-сервера для авторизации.</p> <pre>"ldap" : {   "url" : "ldap://127.0.0.1:389",   "suffix" : "dc=example,dc=com",   "login" : "cn=admin,dc=example,dc=com",   "pass" : "root" }</pre>
ldap.login	Логин администратора LDAP.
ldap.pass	Пароль администратора LDAP.
ldap.suffix	Корневой элемент (как правило, доменное имя организации).
ldap.url	Адрес LDAP-сервера.
MQTT	<p>В этом разделе задаются настройки подключения к MQTT-брокеру.</p> <pre>"mqtt" : {   "broker" : "mqtt://username:password@localhost:1883" }</pre>
mqtt.broker	<p>Адрес и порт брокера.</p> <p>По умолчанию - <b>"mqtt://localhost:1883"</b>.</p> <p><i>Примечание: Для аутентификации по имени пользователя и паролю нужно указать пользовательские данные перед адресом сервера.</i></p>

Раздел/Параметр	Описание
OpenTSDB	В этом разделе задаются параметры доступа к OpenTSDB.
openTsdB.enabled	Запись исторических данных в OpenTSDB; <ul style="list-style-type: none"> <li>• false - запись выключена,</li> <li>• true - запись включена.</li> </ul> По умолчанию - <b>true</b> - запись включена.
openTsdB.host	Адрес хоста с OpenTSDB. По умолчанию - <b>"localhost"</b> .
openTsdB.port	Порт OpenTSDB. По умолчанию - <b>4242</b> .
Push notification	В этом разделе задаются параметры push-уведомлений в мобильном приложении. Для работы с ними используется Firebase Cloud Messaging (FCM).
push_notification.disabled	Выключает уведомления; <ul style="list-style-type: none"> <li>• true - уведомления выключены,</li> <li>• false - уведомления включены.</li> </ul> По умолчанию - <b>true</b> - уведомления выключены.
push_notification.key_path	Путь к ключу авторизации сервера Центрального Пульта на сервере Firebase. По умолчанию - <b>"/etc/saymon/saymon-mobile-firebase-adminsdk.json"</b> .
push_notification.on_state_change	Включает отправку уведомления при изменении состояний на случай другого источника уведомлений - MQTT-сообщений. <ul style="list-style-type: none"> <li>• true - отправка включена,</li> <li>• false - отправка выключена.</li> </ul> По умолчанию - <b>true</b> - отправка включена.
push_notification.timeout	Время, через которое каждому пользователю отправляется уведомление (в миллисекундах). По умолчанию - <b>0</b> - задержки нет.
push_notification.url	URL, полученный пользователем от Firebase, для принятия содержимого уведомлений с сервера. По умолчанию - <b>"https://saymon-mobile.firebaseio.com"</b> .

Раздел/Параметр	Описание
Server	В этом разделе задаются общие параметры сервера.
server.analytics_enabled	<p>Включает аналитику значений метрик в процесс обработки данных;</p> <ul style="list-style-type: none"> <li>• false - аналитика выключена,</li> <li>• true - аналитика включена.</li> </ul> <p>По умолчанию - <b>false</b> - аналитика выключена.</p>
server.analytics_processes	<p>Количество логических ядер, выделяемых для обработки аналитики временных рядов.</p> <p>По умолчанию используются все доступные ядра.</p>
server.colorize_log	<p>Включает цветную раскраску лога;</p> <ul style="list-style-type: none"> <li>• false - раскраска выключена,</li> <li>• true - раскраска включена.</li> </ul> <p>По умолчанию - <b>false</b> - раскраска выключена.</p>
server.comet_ping_interval	<p>Временной интервал между отправками comet-сервером сообщений (в миллисекундах).</p> <p>По умолчанию - <b>5000</b> - 5 секунд.</p>
server.comet_ping_timeout	<p>Время ожидания сообщения от comet-сервера (в миллисекундах).</p> <p>По умолчанию - <b>12000</b> - 12 секунд.</p>
server.comet_port	<p>Порт для соединения.</p> <p>По умолчанию - <b>8091</b>.</p>
server.comet_secure	<p>Включает SSL-соединение;</p> <ul style="list-style-type: none"> <li>• false - соединение выключено,</li> <li>• true - соединение включено.</li> </ul> <p>По умолчанию - <b>false</b> - соединение выключено.</p>
server.comet_ssl_certificate	Путь к сертификату.
server.comet_ssl_key	Путь к ключу.
server.conditional_incidents_enabled	<p>Включает функционал генерации инцидентов;</p> <ul style="list-style-type: none"> <li>• false - функционал выключен,</li> <li>• true - функционал включен.</li> </ul> <p>По умолчанию - <b>false</b> - функционал выключен.</p>

Раздел/Параметр	Описание
server.debug	<p>Включает debug-режим для логирования в файл <b>"/var/log/saymon/saymon-server.log"</b>;</p> <ul style="list-style-type: none"> <li>• false - режим выключен,</li> <li>• true - режим включен.</li> </ul> <p>По умолчанию - <b>false</b> - режим выключен.</p>
server.default_result_timeout	<p>Время, через которое срабатывает условие "Нет данных от объекта" с момента создания объекта или получения последних данных (в миллисекундах).</p> <p>По умолчанию - <b>120000</b> - 2 минуты.</p>
server.default_state_id	<p>Состояние объекта по умолчанию.</p> <p>По умолчанию - <b>7</b> - "Нет данных".</p>
server.discovery_parent_id	<p>ID объекта, в котором появляются найденные агентами сетевые устройства.</p> <p>По умолчанию - <b>"1"</b>.</p>
server.event_log_max_bytes	<p>Размер записей консоли в mongoDB, при достижении которого происходит ротация данных (в байтах).</p> <p>По умолчанию - <b>"1 G"</b> - 1 гигабайт.</p>
server.extension_path	<p>Путь к директории с серверными расширениями.</p>
server.history_temporary_storage_period	<p>Интервал времени для буферизации метрик, по истечении которого все данные из Redis записываются в OpenTSDB (в миллисекундах).</p> <p>Применяется, только если параметр "history_update_period" равен 0.</p>
server.history_update_period	<p>Интервал записи исторических данных (в миллисекундах).</p> <p>0 - немедленная запись пришедших значений.</p> <p>По умолчанию - <b>60000</b> - 1 минута.</p>
server.notification_buffering_period	<p>Период ожидания для сбора сообщений о смене состояний объектов и отправки группового уведомления (в миллисекундах).</p> <p>По умолчанию - <b>0</b> - буферизация отключена.</p>



Раздел/Параметр	Описание
server.retain_expired_stat	Включает хранение последних полученных данных после их устаревания; <ul style="list-style-type: none"> <li>• false - хранение выключено,</li> <li>• true - хранение включено.</li> </ul> По умолчанию - <b>false</b> - хранение выключено.
server.script_trigger_timeout	Максимальное время выполнения триггера (в миллисекундах). По умолчанию - <b>30000</b> - 30 секунд.
server.self_object_id	ID объекта, используемого для самомониторинга.
server.sms_script	Путь до скрипта, отправляющего sms-уведомления.
server.sql_history_enabled	Включает запись исторических данных в MySQL; <ul style="list-style-type: none"> <li>• false - запись выключена,</li> <li>• true - запись включена.</li> </ul> По умолчанию - <b>false</b> - запись выключена.
server.stat_local_timestamp_field_name	Имя поля, где передается время, с которым нужно сохранять данные в OpenTSDB. По умолчанию - <b>"localTimestamp"</b> .
server.stat_scan_period	Период проверки актуальности пришедших данных (в миллисекундах). По умолчанию - <b>3000</b> - 3 секунды.
server.voice_call_script	Путь до скрипта, осуществляющего голосовой вызов.

Раздел/Параметр	Описание
server.email	<p>В этом подразделе задаются параметры доступа к почтовому серверу:</p> <pre> "email" : {   "disabled" : false,   "fields" : {     "from" : "saymon@saas.saymon.info"   },   "max_json_length": 1000,   "transport" : {     "auth" {       "user" : "saymon@saas.saymon.info",       "pass" : "P@ssw0rd"     },     "host" : "smtp.gmail.com",     "port" : 465,     "secure" : true   }, }</pre>
server.email.disabled	<p>Выключает отправку почтовых уведомлений;</p> <ul style="list-style-type: none"> <li>• true - отправка выключена,</li> <li>• false - отправка включена.</li> </ul> <p>По умолчанию - <b>true</b> - отправка выключена.</p>
server.email.fields	Данные об отправителе уведомлений.
server.email.fields.from	Почтовый адрес отправителя.
server.email.max_json_length	<p>Ограничение размера письма с уведомлением (в символах).</p> <p>По умолчанию - <b>1000</b> - 1000 символов.</p>
server.email.transport	Данные почтового сервера.
server.email.transport.auth	Данные для аутентификации пользователя.
server.email.transport.auth.pass	Пароль пользователя.
server.email.transport.auth.user	Логин пользователя.
server.email.transport.host	Адрес почтового сервера.
server.email.transport.port	Порт почтового сервера.

Раздел/Параметр	Описание
server.email.transport.secure	Включает использование TLS при подключении к серверу; <ul style="list-style-type: none"> <li>• false - использование TLS выключено,</li> <li>• true - использование TLS включено.</li> </ul> Значение по умолчанию зависит от порта.
server.email.transport.service	Встроенный в коннектор набор служб. При наличии задаёт "host", "port", "secure" автоматически. По умолчанию - <b>"Gmail"</b> .
server.user	В этом подразделе задаются параметры пользователей: <pre> "user" : {   "auth_enabled": "true",   "new_user_access": "all",   "lang_default": "ru",   "template": {     "permissions": ["manage-objects", "view-section-stat"],     "objectPermissions": {       "include": [],       "exclude": ["5fb643ddf277b96c8401119b"]     }   },   "usersRoot": "5800d9aaac7bf0f90d3d520e" } </pre>
server.user.auth_enabled	Включает самостоятельную регистрацию для пользователей; <ul style="list-style-type: none"> <li>• false - регистрация выключена,</li> <li>• true - регистрация включена.</li> </ul> По умолчанию - <b>false</b> - регистрация выключена.
server.user.lang_default	Язык пользователей по умолчанию; <ul style="list-style-type: none"> <li>• "en" - английский,</li> <li>• "it" - итальянский,</li> <li>• "ru" - русский.</li> </ul> По умолчанию - <b>"en"</b> - английский.
server.user.new_user_access	Права доступа к объектам для нового пользователя; <ul style="list-style-type: none"> <li>• "all" - есть доступ ко всем объектам,</li> <li>• "not" - нет доступа ни к одному объекту.</li> </ul> По умолчанию - <b>"all"</b> - доступ ко всем объектам.

Раздел/Параметр	Описание
server.user.template	Шаблон прав нового пользователя. <i>Примечание: Данные параметры применяются только для пользователей, регистрирующихся самостоятельно.</i>
server.user.template.objectPermissions	Права пользователя на доступ к объектам.
server.user.template.objectPermissions.include	Список идентификаторов объектов, к которым пользователю по умолчанию доступ разрешён.
server.user.template.objectPermissions.exclude	Список идентификаторов объектов, к которым пользователю по умолчанию доступ запрещён.
server.user.template.permissions	Список прав на операции, доступных пользователю по умолчанию.
server.user.usersRoot	Идентификатор корневого объекта для создаваемых новым пользователем объектов. По умолчанию - "1".
Resource_server	В этом разделе задаются параметры, связанные с хранением файлов, загруженных в Центральный Пульт.
resource.server.debug	Включает debug-режим для логирования в файл <b>"/var/log/saymon/saymon-server.log"</b> ; <ul style="list-style-type: none"> <li>• false - режим выключен,</li> <li>• true - режим включен.</li> </ul> По умолчанию - <b>false</b> - режим выключен.
resource.server.file_storage_dir	Путь к директории для хранения документов, прикрепляемых к объектам. По умолчанию - <b>"/var/saymon/resources"</b> .
resource.server.ip_address	Адрес Resource-сервера. По умолчанию - <b>"127.0.0.1"</b> .
resource.server.port	Порт Resource-сервера. По умолчанию - <b>8092</b> .

Раздел/Параметр	Описание
Rest_server	В этом разделе задаются параметры REST-сервера.
rest.server.base_url	Путь к API. По умолчанию - <b>"/api"</b> .
rest.server.colorize_log	Включает цветную раскраску лога; <ul style="list-style-type: none"> <li>• false - раскраска выключена,</li> <li>• true - раскраска включена.</li> </ul> По умолчанию - <b>false</b> - раскраска выключена.
rest.server.debug	Включает debug-режим для логирования в файл <b>"/var/log/saymon/saymon-server.log"</b> ; <ul style="list-style-type: none"> <li>• false - режим выключен,</li> <li>• true - режим включен.</li> </ul> По умолчанию - <b>false</b> - режим выключен.
rest.server.document_download_url	URL к файлам, сохраненным в <b>\$document_storage_dir</b> . По умолчанию - <b>"http://localhost/node/api/docs"</b> .
rest.server.ip_address	Адрес хоста для запуска REST-сервера. По умолчанию - <b>"127.0.0.1"</b> .
rest.server.num_workers	Число процессов для загрузки данных. По умолчанию - <b>1</b> .
rest.server.port	Порт REST-сервера. По умолчанию - <b>8090</b> .
rest.server.public_url	Адрес для доступа к web-интерфейсу из уведомлений.
rest.server.update_download_url	Путь к файлу для обновления агента. По умолчанию - <b>"http://localhost/node/api/agents/update"</b> .

### 5.1.4 Опции конфигурации

Управление некоторыми настройками доступно в web-интерфейсе системы.

Изменённые параметры в интерфейсе приоритетнее настроек в конфигурационном файле.

Установленные значения параметров автоматически сохраняются в MongoDB, конфигурационный файл не перезаписывается.

Раздел "Опции конфигурации" окна конфигурации системы состоит из:

1. "Общие настройки" соответствуют параметрам раздела "Server" (Рис. 4):

Общие настройки	
Сохранять устаревшие данные от агента	<input type="radio"/> НЕТ <input checked="" type="radio"/>
Период обновления истории, мс	120000
Период обновления таблицы состояний, мс	
Период проверки устаревания данных от агента, мс	
Период обработки пассивных данных от агента, мс	1000
Максимальный размер журнала событий, байты	1073741824
Срок действия данных от агента по умолчанию, мс	2000
Таймаут погашенной аварии, мс	10000

Рис. 4. Общие настройки

2. "Настройки электронной почты" соответствуют параметрам подраздела "Email" (Рис. 5):

Настройки электронной почты	
Выключено	<input type="radio"/> НЕТ <input checked="" type="radio"/>
Сервис транспорта	smtp.yandex.ru
Имя пользователя	saymon@saymon.info
Пароль	.....
Адрес отправителя	saymon@saymon.info
Адрес получателя	
Максимальная длина JSON	100000

Рис. 5. Настройки электронной почты

3. "Настройки пользователей" соответствуют параметрам подраздела "User" (Рис. 6):

Рис. 6. Настройки пользователей

Просмотреть актуальные настройки сервера возможно при помощи REST API метода:

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

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
url=https://$saymon_hostname/node/api/configuration

curl -X GET $url -u $login:$password
```

## 5.15 Служба "saymon-server"

Узнать состояние, запустить, перезапустить и остановить службу сервера можно следующими командами соответственно:

```
sudo service saymon-server status  
sudo service saymon-server start  
sudo service saymon-server restart  
sudo service saymon-server stop
```



## 5.16 Просмотр websocket-нотификаций

Центральный пульт использует гибкий механизм оповещений, который позволяет пользователю оперативно реагировать на возникающие ситуации.

Чтобы просмотреть пришедшие websocket-уведомления, необходимо:

1. Запустить Chrome;
2. Авторизоваться на сервере Центрального Пульта;
3. Перейти по ссылке `https://<your_server>/incidents.html?debug=comet`
4. Открыть в Chrome Инструменты разработчика;
5. Открыть вкладку Консоль (Console) в Инструментах разработчика;
6. В качестве примера нажать правой кнопкой мыши на строке с инцидентом и выбрать пункт "Подтвердить".

В консоли отобразится websocket-нотификация.

### 5.17 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 возможно увеличить количество 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 необходимо выполнить команду:

```
ps ax | grep node
```

В выводе должно отображаться то количество процессов "SnmpTrapMessageHandlerActor", которое было указано в параметре "clusterSize" выше (в этом примере - 3):

```
5793 ?   Rsl  10:23 /usr/bin/nodejs --  
harmony /usr/local/saymon/backend/server/saymon-server.js  
  
5857 ?   Sl   3:18 /opt/nodejs/bin/node --  
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js RestServerActor  
  
5862 ?   Sl   0:37 /opt/nodejs/bin/node --  
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js ResourceServerActor  
  
5867 ?   Sl   1:43 /opt/nodejs/bin/node --  
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js HistoryWriterActor  
  
5872 ?   Sl   0:45 /opt/nodejs/bin/node --  
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js  
SnmpTrapMessageHandlerActor  
  
5877 ?   Sl   0:42 /opt/nodejs/bin/node --  
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js  
SnmpTrapMessageHandlerActor  
  
5882 ?   Sl   0:42 /opt/nodejs/bin/node --  
harmony /usr/local/saymon/backend/server/actors/forked-actor-worker.js  
SnmpTrapMessageHandlerActor
```

### 5.18 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
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 хранятся в папках:

- Linux, Wiren Board 6 - /var/log/saymon;
- Mac OS X - /opt/saymon-agent/log;
- Windows - папка\_установки\_агента\log.

Также в журналы записывается информация о следующих событиях:

- дата и время удачных и неудачных попыток входа пользователей в систему;
- причина и время выхода пользователей из системы;
- дата, время и инициатор выполнения операций;
- факт использования прав администратора;
- факт доступа пользователей к основным конфигурационным данным платформы;
- запуск и остановка сервисов аудита действий пользователя.

Все записи имеют временной штамп, признак инициатора (ID пользователя или процесса) и признак сессии, если событие было инициировано пользователем.

### 5.2.1 Конфигурация log-файлов

Конфигурация log-файлов агента осуществляется в файле **"/opt/saymon-agent/conf/logback-upstart.xml"**.

Секция настройки debug-режима:

```
<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:

Параметр	Описание
/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 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, уреза 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:

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

Пример (bash):


```
login=<...>
password=<...>
saymon_hostname=<...>
url=https://$saymon_hostname/node/api/event-log/info

curl -X GET $url -u $login:$password
```

## 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 открыть Журнал событий кнопкой .
- 1.2. Нажать правой кнопкой мыши на требуемое событие, затем выбрать пункт "Назначить ответственного".
- 1.3. Выбрать ответственного за событие из выпадающего списка.

Имя ответственного пользователя отобразится в соответствующем столбце (Рис. 7):

STAGING Журнал Событий <span>  </span> SNMP <span>▼</span> <span>▼</span> Фильтр <span>▼</span> Вкл Количество сообщений: 100								
Количество <span>↕</span>	Время <span>▼</span>	Критичность <span>↕</span>	Объект на схеме <span>↕</span>	Адрес отправителя <span>↕</span>	OID трапа <span>↕</span>	Текст <span>↕</span>	Данные <span>↕</span>	Ответственный <span>↕</span>
100	03.11.2020, 13:40:53	Major	IT'S A TRAP	127.0.0.1	.1.3.6.1.4.1.5089.2.0.99	0	.1.3.6.1.4.1.5089.2.0.99 "0"	admin
	03.11.2020, 13:39:53	Major	IT'S A TRAP	127.0.0.1	.1.3.6.1.4.1.5089.2.0.99	0	.1.3.6.1.4.1.5089.2.0.99 "0"	
	03.11.2020, 13:37:53	Major	IT'S A TRAP	127.0.0.1	.1.3.6.1.4.1.5089.2.0.99	0	.1.3.6.1.4.1.5089.2.0.99 "0"	

Рисунок. 7. Журнал событий

### 2. REST API методом:

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

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
record_id=<...>
url=https://$saymon_hostname/node/api/event-log/$record_id/assignee

curl -X PATCH $url -u $login:$password -H "Content-Type: application/json" \
--data '{"userId": "..."}'
```



### 5.2.5 Установка ограничения для логирования

Центральный Пульт позволяет установить максимальный объем хранилища двумя способами:

1. Через Web UI:

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

1.2. Перейти в раздел "Журнал событий" (Рис. 8):

Рис. 8. Ограничение объема хранилища

1.3. Заполнить поля требуемыми значениями.

1.4. Сохранить изменения.

2. Через REST API метод:

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

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
url=https://$saymon_hostname/node/api/event-log/limits

curl -X PUT $url -u $login:$password -H "Content-Type: application/json" \
--data '{"maxBytes": 1024, "maxRecords": 100}'
```

### 5.2.6 Удаление всех SNMP-Trap'ов из Журнала событий

Для очистки Журнала событий необходимо:

1. Выполнить в терминале следующие команды:

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

2. После сброса проверить создание новой коллекции командой:

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

3. Если ответ выглядит следующим образом, необходимо создать новую коллекцию:

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

- создание новой коллекции с лимитом по объёму в байтах:

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

- создание новой коллекции без лимита:

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

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/*
```

Также возможно удалить логи с помощью REST API:

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

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
record_id=<...>
url=https://$saymon_hostname/node/api/event-log/$record_id

curl -X DELETE $url -u $login:$password
```

## 5.3 Управление учётными записями пользователей

Раздел содержит информацию об администрировании пользовательских учётных записей.

## 5.3.1 Создание учётных записей

Создание учётных записей пользователей осуществляется двумя способами:

1. Через Web UI:

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

1.2. Перейти в раздел "Пользователи" (Рис. 9):

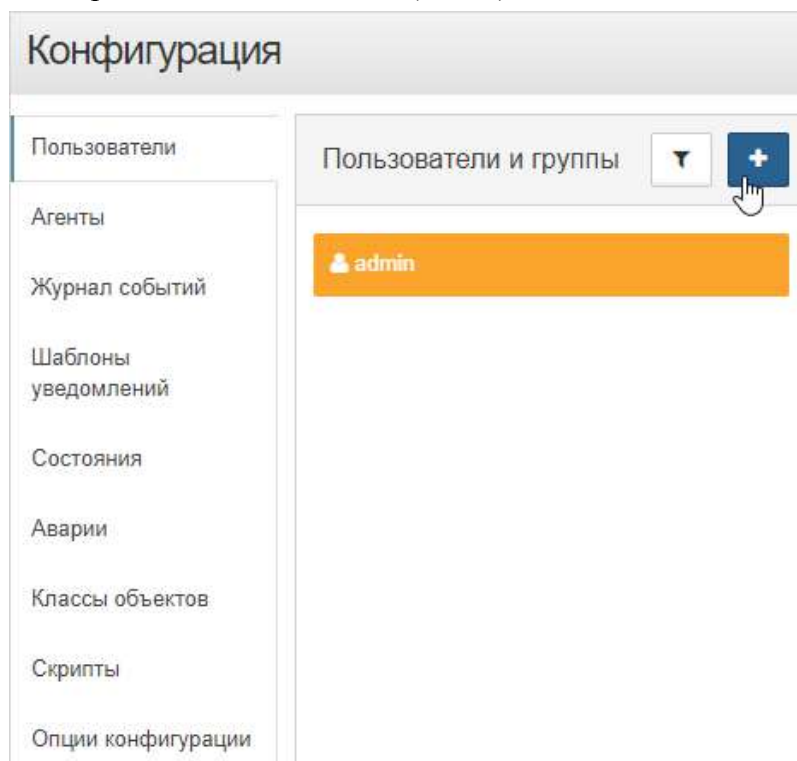



Рис. 9. Добавление пользователя

1.3. Нажать кнопку  и выбрать добавление пользователя.

1.4. Ввести логин пользователя, пароль и подтверждение.

1.5. Нажать кнопку  .

## 2. Через REST API:

```
POST /node/api/users
```

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
url=https://$saymon_hostname/node/api/users

curl -X POST $url -u $login:$password -H "Content-Type: application/json" -d @- <<EOF
{
    "login": "Bob",
    "password": "qwerty",
    "permissions": [
        "manage-objects",
        "manage-links"
    ]
}
EOF
```

### 5.3.2 Назначение пользователям прав доступа

Настройка прав пользователей осуществляется двумя способами:

1. Через Web UI:

- 1.1. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
- 1.2. Перейти в раздел "Пользователи".
- 1.3. Выбрать нужного пользователя из списка.
- 1.4. Открыть вкладку "Права на операции" (Рис. 10):

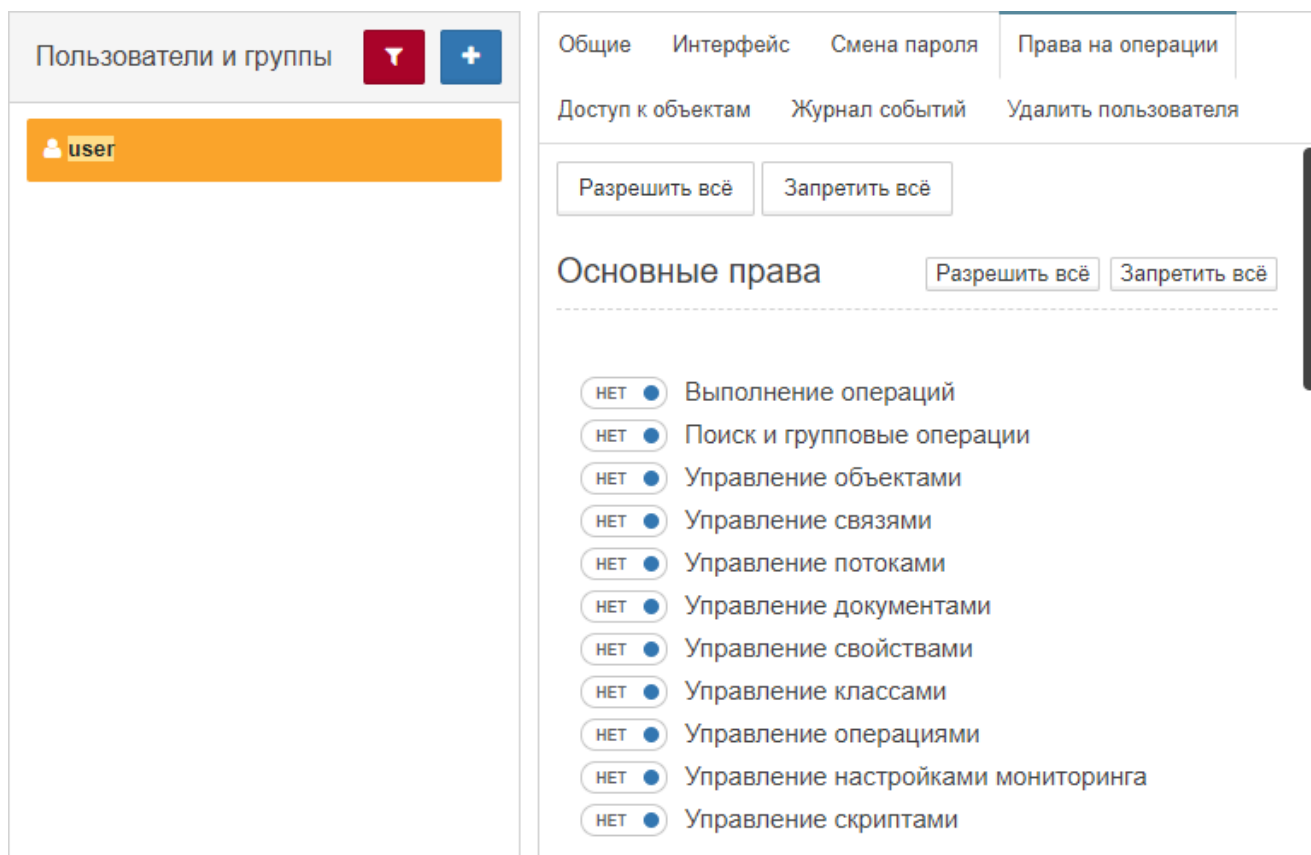


Рис. 10. Права на операции

- 1.5. Отредактировать права пользователя.

## 2. Через REST API:

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

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
user_id=<...>
url=https://$saymon_hostname/node/api/users/$user_id

curl -X PATCH $url -u $login:$password -H "Content-Type: application/json" -d @- <<EOF
{
    "login": "Bob",
    "password": "qwerty",
    "permissions": [
        "manage-objects",
        "manage-links"
    ]
}
EOF
```



### 5.3.3 Изменение пароля от учётной записи

Смена пароля пользователей осуществляется двумя способами:

1. Через Web UI:

- 1.1. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
- 1.2. Перейти в раздел "Пользователи".
- 1.3. Выбрать нужного пользователя из списка.
- 1.4. Открыть вкладку "Смена пароля" (Рис. 11):

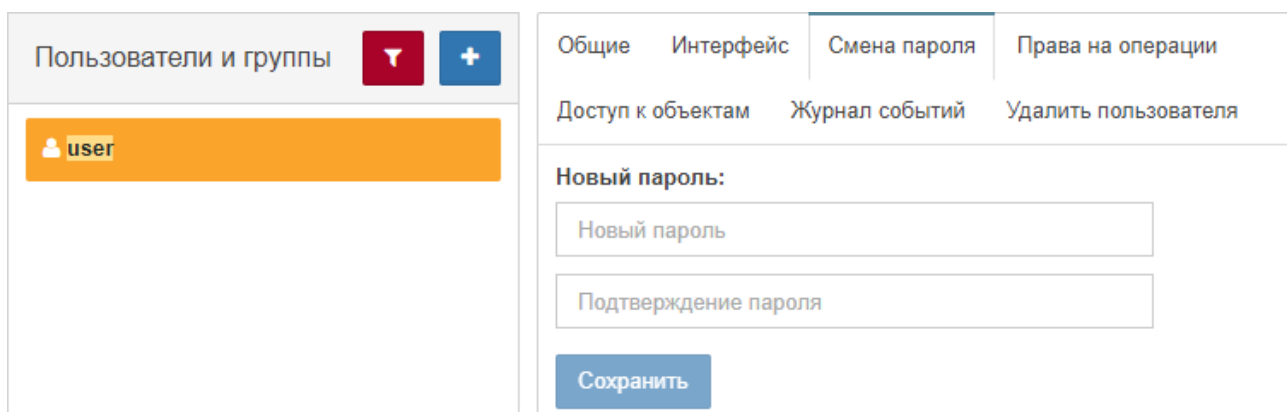


Рис. 11. Смена пароля пользователя

- 1.5. Ввести новый пароль пользователя и подтверждение.

*Примечание: При смене собственного пароля необходимо также ввести текущий пароль.*

- 1.6. Нажать кнопку .

2. Через REST API:

```
PUT /node/api/users/:id/password
```

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
user_id=<...>
url=https://$saymon_hostname/node/api/users/$user_id/password

curl -X PUT $url -u $login:$password -H "Content-Type: application/json" -d @- <<EOF
{
  "currentPassword": "qwerty",
  "newPassword": "qwerty_qwerty"
}
EOF
```

### 5.3.4 Удаление пользователя

Удаление пользователей осуществляется двумя способами:

1. Через Web UI:

- 1.1. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
- 1.2. Перейти в раздел "Пользователи".
- 1.3. Выбрать нужного пользователя из списка.
- 1.4. Открыть вкладку "Удалить пользователя" (Рис. 12):

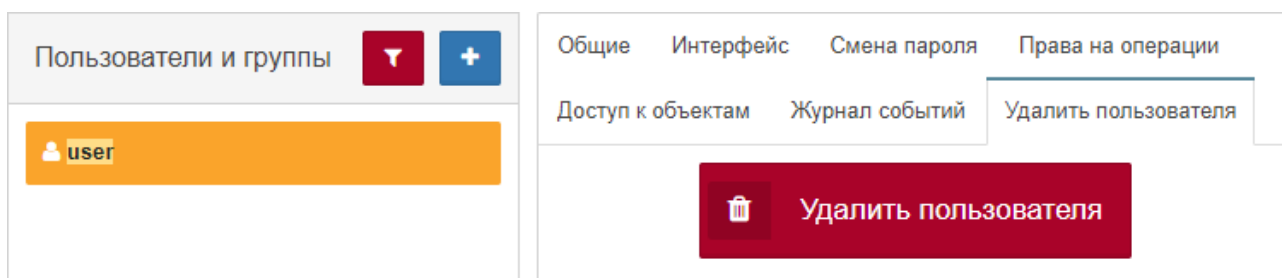


Рис. 12. Удаление пользователя

- 1.5. Нажать кнопку удаления пользователя.
- 1.6. Подтвердить удаление во всплывающем окне.

2. Через REST API:

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

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
user_id=<...>
url=http://$saymon_hostname/node/api/users/$user_id

curl -X DELETE $url -u $login:$password
```

## 5.4 Работа с объектами и связями

Вся управляемая инфраструктура в системе представлена в виде объектов и связей между ними.

Все метрики, характеризующие текущее состояние платформы, связываются с отдельными объектами мониторинга. Контроль загрузки и наличия свободных ресурсов осуществляется стандартными средствами платформы.

### 5.4.1 Создание объекта

Создать объект можно двумя способами:

1. Через Web UI:

1.1. Нажать кнопку  на панели инструментов.

1.2. В окне "Новый объект" ввести имя объекта и выбрать класс объекта (Рис. 13):

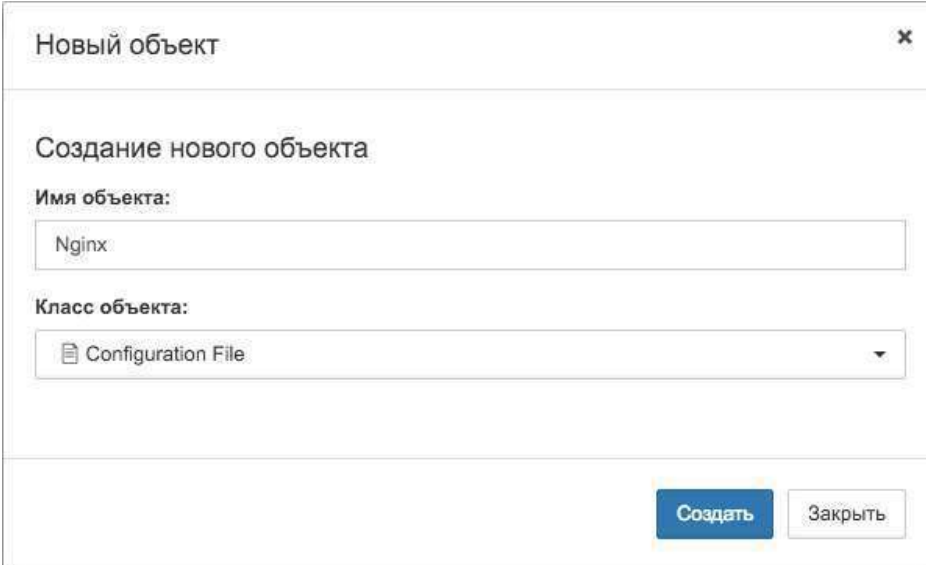


Рис. 13. Окно создания нового объекта

1.3. Нажать кнопку .

2. Через REST API:

POST /node/api/objects

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
url=https://$saymon_hostname/node/api/objects

curl -X POST $url -u $login:$password -H "Content-Type: application/json" -d @- <<EOF
{
    "name": "New Object",
    "parent_id": "1"
    "class_id": "3"
}
EOF
```

Примечание: "parent\_id" - ID родительского объекта для создаваемого

## 5.4.2 Клонирование объекта

Объект клонируется со всеми своими документами, параметрами, свойствами, дочерними объектами и связями.

Клонировать объект можно двумя способами:

1. Через Web UI:

- 1.1. Вызвать контекстное меню клонируемого объекта щелчком правой кнопкой мыши.
- 1.2. Выбрать пункт меню "Клонировать".

2. Через REST API:

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

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
object_id=<...>
url=https://$saymon_hostname/node/api/objects/$object_id/clone

curl -X POST $url -u $login:$password
```


### 5.4.3 Удаление объекта

Удалить объект можно тремя способами:

1. Через Web UI (контекстное меню объекта):

- 1.1. Вызвать контекстное меню удаляемого объекта щелчком правой кнопкой мыши.
- 1.2. Выбрать пункт меню "Удалить".
- 1.3. Подтвердить удаление объекта во всплывающем окне.

2. Через Web UI (режим удаления элементов):

- 2.1. Нажать кнопку  в панели хлебных крошек.
- 2.2. Нажать на такую же иконку на удаляемом объекте.
- 2.3. Подтвердить удаление объекта во всплывающем окне.

3. Через REST API:

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

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
object_id=<...>
url=https://$saymon_hostname/node/api/objects/$object_id

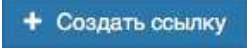
curl -X DELETE $url -u $login:$password
```

## 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, в дашбордах.

Создать ссылку можно двумя способами:

### 1. Через Web UI:

- 1.1. Нажать на кнопку  на панели инструментов.
- 1.2. В появившемся всплывающем окне "Новая ссылка" (Рис. 14) выбрать из выпадающего списка объект, на который создаётся ссылка:

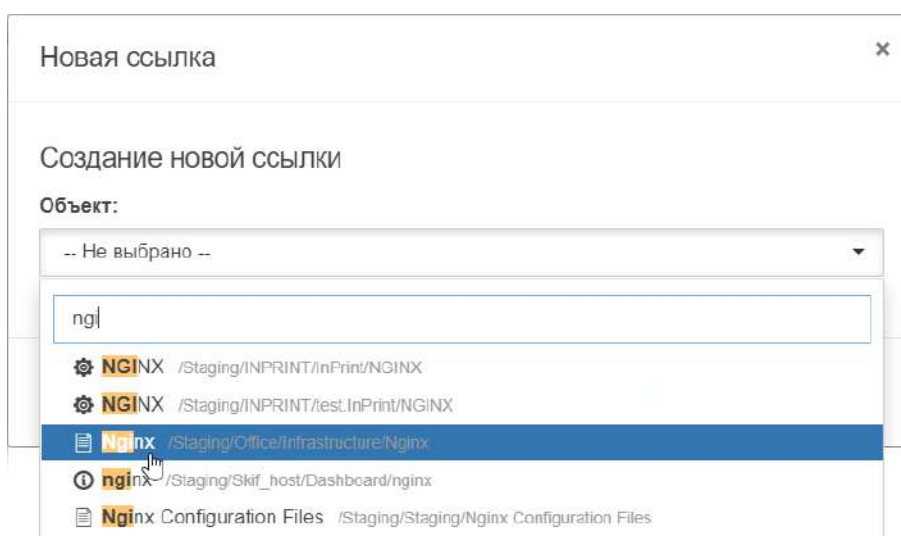


Рис. 14. Окно создания новой ссылки

- 1.3. Нажать на кнопку .

### 2. Через REST API:

POST /node/api/refs

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
url=https://$saymon_hostname/node/api/refs


curl -X POST $url -u $login:$password -H "Content-Type: application/json" \
--data '{"target": "5e60d9db630502472925fe9f", "owner": "1"}'
```

## 5.4.5 Создание связи

Связи между объектами также могут являться объектами мониторинга.

Создать связь можно двумя способами:

### 1. Через Web UI:

- 1.1. Перейти в режим создания связей, нажав кнопку **+ Создать связь** на панели инструментов.
- 1.2. После того, как на всех объектах появится соответствующий символ , нажать на него на исходном объекте и, удерживая, переместить курсор на целевой объект (Рис. 15):

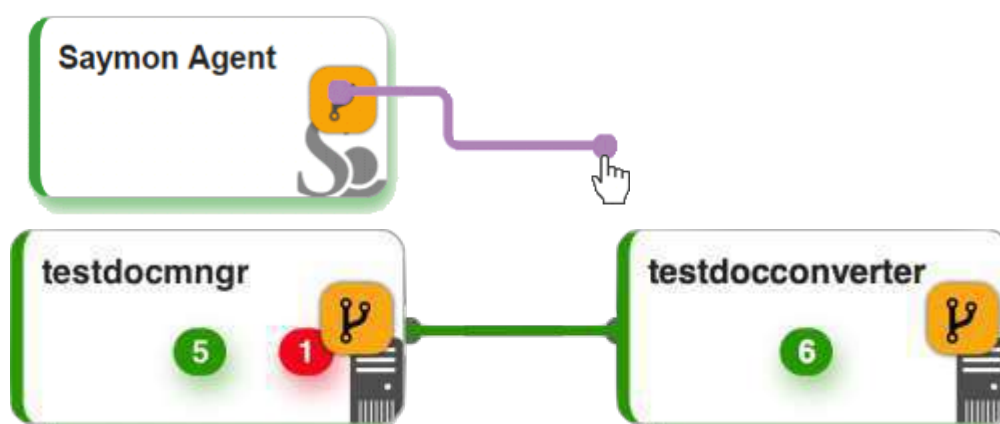


Рис. 15. Создание связи

- 1.3. Нажать кнопку **Выйти** в верхней части главного экрана для выхода из режима создания связей.

### 2. Через REST API:

POST /node/api/links

Пример (bash):

```
login=<...>
password=<...>
saymon_hostname=<...>
source_obj_id=<...>
target_obj_id=<...>
url=https://$saymon_hostname/node/api/links


curl -X POST $url -u $login:$password -H "Content-Type: application/json" -d @- <<EOF
{
  "source": "$source_obj_id",
  "target": "$target_obj_id"
}
EOF
```



## 5.4.6 Удаление связи

Удалить связь можно двумя способами:

### 1. Через Web UI:

- 1.1. Нажать кнопку  в панели хлебных крошек.
- 1.2. Нажать на такую же иконку на удаляемой связи.
- 1.3. Подтвердить удаление связи во всплывающем окне.

### 2. Через REST API:

```
DELETE /node/api/links/:id
```

Пример (bash):

```
login=<...>  
password=<...>  
saymon_hostname=<...>  
link_id=<...>  
url=https://$saymon_hostname/node/api/links/$link_id  
  
curl -X DELETE $url -u $login:$password
```

## 5.5 Настройка уведомлений

При переходе объектов в определенные состояния система может:

- 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 настраиваются в виде подробной информации об объекте в секции "Действия при смене состояния" (Рис. 16):

Действия при смене состояния

Email	duty_operator@saymon.info	admin@saymon.info	Working Alarm Overloaded			
Программа / скрипт	/reboot.sh	Аргументы	Alarm			
Telegram	123456:ABCD	42	Working			
Звуковое уведомление	admin	sound1	Alarm Overloaded			
Задача в Jira	Alarm					
Операция	MQTT Message	Alarm Overloaded				
SMS уведомление	+7 555 1234567	Alarm				
Голосовое уведомление	+7 555 7654321	Alarm				

+ Добавить триггер

Рис. 16. Действия при смене состояния

### 5.5.1 Активация функционала отправки SMS и голосовых вызовов

Для активации функционала необходимо выполнить описанные ниже действия:

1. В файле конфигурации клиента **"/usr/local/saymon/client/client-config.js"** установить параметрам "enableSmsTrigger" и "enableVoiceCallTrigger" значения "true":

```
return {  
  ...  
  enableVoiceCallTrigger : true,  
  enableSmsTrigger : true,  
  ...  
}
```

2. В разделе "Server" конфигурационного файла сервера **"/etc/saymon/saymon-server.conf"** задать путь до скриптов, осуществляющих от отправку SMS-уведомлений и голосовые вызовы:

```
"server" : {  
  ...  
  "sms_script" : "путь до скрипта, отправляющего sms-уведомления",  
  "voice_call_script" : "путь до скрипта, осуществляющего голосовые вызовы",  
  ...  
}
```

*Примечание: Скрипт, отправляющий SMS-уведомления, получает в качестве аргументов:*

*\$1 - номер телефона получателя (как введен в триггере);  
\$2 - ID объекта, в котором сработал триггер;  
\$3 - имя объекта, в котором сработал триггер;  
\$4 - ID состояния объекта;  
\$5 - текст уведомления, настроенного в шаблонах уведомлений.*

*Скрипт, осуществляющий голосовой вызов, получает в качестве аргументов:*

*\$1 - номер телефона получателя (как введен в триггере);  
\$2 - ID объекта, в котором сработал триггер;  
\$3 - имя объекта, в котором сработал триггер;  
\$4 - ID состояния объекта.*

3. Для применения изменений перезапустить службу "saymon-server":

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

## 5.5.2 Отправка почтовых уведомлений

Для настройки отправки почтовых уведомлений требуется настроить необходимые параметры доступа к почтовому серверу в подразделе "Email" раздела "Server" файла конфигурации сервера `/etc/saymon/saymon-server.conf`.

Подробности см. в разделе 5.1.3 "Конфигурация сервера".

### 5.5.3 Настройка уведомлений в Telegram

Для настройки отправки уведомлений с помощью Telegram необходимо:

#### 1. Задать бота, от которого будут поступать уведомления:

##### 1.1. Создать нового бота:

- 1.1.1. Найти в приложении Telegram контакт "BotFather".
- 1.1.2. Отправить ему сообщение "/newbot".
- 1.1.3. Задать боту отображаемое имя (name - позже возможно изменить).
- 1.1.4. Задать боту уникальное имя (username - изменить будет невозможно).
- 1.1.5. Скопировать токен бота вида  
`210979209:AAFfT2mt3oW4EK1gYqE_d3OjAJSIRLSrAL`
- 1.1.6. Отправить контакту "BotFather" сообщение "/setprivacy".
- 1.1.7. Выбрать созданного бота по его username.
- 1.1.8. Выбрать опцию Disable.

##### 1.2. Использовать существующего бота:

- 1.2.1. Найти в приложении Telegram контакт "BotFather".
- 1.2.2. Отправить ему сообщение "/mybots".
- 1.2.3. Выбрать нужного бота.
- 1.2.4. Выбрать опцию API Token.
- 1.2.5. Скопировать токен бота вида  
`210979209:AAFfT2mt3oW4EK1gYqE_d3OjAJSIRLSrAL`

#### 2. Настроить канал, чат или группу:

##### 2.1. Создать приватный канал (рекомендуется):

- 2.1.1. В приложении Telegram создать новый канал.
- 2.1.2. Открыть настройки канала и добавить бота в список администраторов.
- 2.1.3. Отправить сообщение в канал.
- 2.1.4. Перейти по ссылке в любом web-браузере, вставив в неё токен своего бота (без пробелов и знаков <>):  
`https://api.telegram.org/bot<токен_бота>/getUpdates`
- 2.1.5. Найти текст со словами "chat" и "id", например  
... "channel\_post": {"message\_id": 4, "chat": {"id": -1001156346945, "title": "SAYMON" ...  
Здесь -1001156346945 - искомый ID канала.

##### 2.2. Создать чат:

- 2.2.1. Отправить боту любое сообщение.
- 2.2.2. Перейти по ссылке в любом web-браузере, вставив в неё токен своего бота (без пробелов и знаков <>):  
`https://api.telegram.org/bot<токен_бота>/getUpdates`
- 2.2.3. Найти текст со словами "chat" и "id", например  
... ": "К"}, "chat": {"id": 121399918, "first\_ ...  
Здесь 121399918 - искомый ID чата.

## 2.3. Настроить группу:

2.3.1. Добавить бота в группу.

2.3.2. Отправить боту в группу любое сообщение, начав его со знака @.

2.3.3. Перейти по ссылке в любом web-браузере, вставив в неё токен своего бота (без пробелов и знаков < >):

`https://api.telegram.org/bot<токен_бота>/getUpdates`

2.3.4. Найти текст со словами "chat" и "id", например

`... {"k": "К"}, {"chat": {"id": -209194473, "first_ ...`

Здесь -209194473 - искомый ID группы.


3. В настройках Telegram-уведомлений ввести токен (ID) бота и ID канала/чата/группы в соответствующие поля.

## 5.6 Настройка интерфейса

В разделе описаны механизмы настройки элементов интерфейса для лучшей визуализации данных.

## 5.6.1 Выравнивание/расстановка объектов в стандартном виде

Выравнивание объектов в стандартном виде происходит благодаря сетке.

Для отображения/скрытия сетки необходимо нажать кнопку  на панели "хлебных крошек".

Настройки сетки выполняется в файле **"/usr/local/saymon/client/client-config.js"**.

В секции "Grid" этого файла можно задать параметры:

- border - максимальный отступ границы объекта от границы сетки (в пикселях).
- color - цвет сетки в формате RGBA.
- dim - размер сетки (в пикселях).

```
grid: {  
  dim: 20,  
  color: "rgba(128, 128, 128, 0.3)",  
  border: 4  
},  
...
```

После внесения изменений в файл обновить страницу в браузере.



## 5.6.2 Настройка заголовка web-интерфейса

Для изменения заголовка в web-интерфейсе:

1. Открыть файл `"/usr/local/saymon/client/client-config.js"`.
2. В строке "title" ввести желаемое имя:

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

3. Обновить страницу браузера.

### 5.6.3 Перемещение/отключение фоновой иконки объекта

Все предустановленные классы объектов имеют индивидуальную фоновую иконку.

При необходимости переместить иконку, в секции "Параметры" во вкладке "Стили" объекта необходимо 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  
  
}
```

## 5.6.4 Вертикальное отображение имени объекта

По умолчанию имя объекта отображается в его левом верхнем углу горизонтально.

При необходимости отображать имя вертикально, в секции "Параметры" во вкладке "Стили" объекта необходимо добавить:

```
.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.

Каждое состояние имеет цвет. Цвет и стиль состояний можно изменить двумя способами:

1. Через файл конфигурации:

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 состояния или 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;  
}  
  
.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 выбрать в меню 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 при помощи цветовой палитры или методом ввода номера цвета.

## 5.7 Настройка мониторинга

"Центральный Пульт" позволяет осуществлять мониторинг с использованием различных типов проверок. Проверки настраиваются в web-интерфейсе системы.

Подробную информацию о каждом типе проверок см. в "Руководстве пользователя".

Далее 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 web-интерфейсе.
3. Перейти в созданный объект и добавить внутри него объекты классов:
  - Saymon Agent,
  - CPU,
  - File System,
  - Memory,
  - Network IO.
4. Сконфигурировать и запустить агента.

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

## 5.7.2 Мониторинг процесса памяти

Для настройки мониторинга процесса памяти необходимо:

1. Установить, сконфигурировать и запустить агента на наблюдаемом ПК или сервере.
2. Создать объект, например, класса "Process", в web-интерфейсе.
3. Перейти в созданный объект и в его секции "Мониторинг":
  - выбрать агента, установленного ранее на данный компьютер;
  - Выбрать тип проверки "Процесс по имени";
  - заполнить необходимые поля.

Через некоторое время информация о проверяемом процессе начнёт поступать на сервер и отображаться в web-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, например, класса "Configuration File", в web-интерфейсе.
3. Перейти в созданный объект и в его секции "Мониторинг":
  - выбрать агента, установленного ранее на данный компьютер;
  - Выбрать тип проверки "Конфигурационный файл / директория";
  - указать путь к проверяемому файлу / директории.

Через некоторое время информация о проверяемом файле / директории начнёт поступать на сервер и отображаться в web-интерфейсе системы.



## 5.7.4 Проверка доступности web-ресурса

Данный тип мониторинга позволяет убедиться не только в работоспособности web-сайта (статус 200 OK), но и в ограничении 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 web-ресурса необходимо:

1. Установить, сконфигурировать и запустить хотя бы одного агента в инфраструктуре.
2. Создать объект, например, класса "Address", в web-интерфейсе.
3. Перейти в созданный объект и в его секции "Мониторинг":
  - выбрать агента, который будет выполнять проверку;
  - Выбрать тип проверки "HTTP-запрос";
  - выбрать тип запроса "GET";
  - в поле "URL" указать адрес web-сайта.

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

### 5.7.5 Безагентный мониторинг web-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, собирать необходимые данные и генерировать текстовый файл с результатами в формате JSON 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
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 в инфраструктуре.
4. Создать объект, например, класса "Info", в web-интерфейсе.
5. Перейти в созданный объект и в его секции "Мониторинг":
  - выбрать агента, который будет выполнять проверку;
  - Выбрать тип проверки "HTTP-запрос";
  - выбрать тип запроса "GET";
  - в поле "URL" указать адрес JSON-файла.

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

## 5.8 Импорт и экспорт данных

Система предусматривает возможность создания резервных копий, восстановления и переноса основных данных между различными инсталляциями.

### Экспорт данных

Для экспорта данных из MongoDB в архив на существующей инсталляции системы необходимо выполнить следующую команду:

```
mongodump -d saymon -o mongodb_backup
```

Для экспорта настроек конфигурации необходимо сделать копии:

- директории с конфигурационными файлами сервера **"/etc/saymon"**;
- конфигурационного файла клиента **"/usr/local/saymon/client/client-config.js"**.

*Примечание: В некоторых случаях файл **client-config.js** может находиться в другой директории, путь к которой может быть найден в конфигурационном файле **nginx** или командой:*

```
sudo find / -name "client-config.js"
```

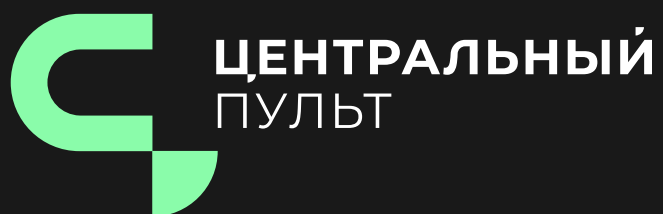
### Импорт данных

Для восстановления или импорта данных в MongoDB на новой инсталляции системы необходимо выполнить следующую команду:

```
mongorestore --drop --noIndexRestore mongodb_backup  
echo flushall | redis-cli
```

Для импорта настроек конфигурации необходимо вставить:

- файлы из ранее сохранённой директории с конфигурационными файлами сервера в директорию **"/etc/saymon"**;
- конфигурационный файл клиента в директорию **"/usr/local/saymon/client"**.



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

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

Раздел содержит информацию об известных проблемах, которые могут возникать при работе с платформой, и способы их решения.

## 6.1 Недостаточно места на виртуальной машине с сервером

### Условия проблемы:

- есть виртуальная машина с сервером Центрального Пульта;
- нет места на виртуальной машине с сервером Центрального Пульта.

### 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)

1. 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
```

2. На хосте с сервером (конфигурация Центрального Пульта):

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

3. 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

1. `less /var/log/opentsdb/opentsdb.log` (здесь можно увидеть какие-то ошибки)
2. `sudo service opentsdb stop`
3. `sudo service hbase restart`
4. `sudo service opentsdb start`

## 6.7 Ошибка работы HTTP-проверки

### Условие проблемы:

- 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.

[www.saymon.info](http://www.saymon.info)

2021

