ЦЕНТРАЛЬНЫЙ ПУЛЬТ



Руководство для администрирования

Содержание

1. Введение 7
   1.1. Описание документа 7
   1.2. Назначение системы 7
2. Системные требования 7
   2.1. Системные требования к серверу 8
   2.2. Системные требования к агенту 8
   2.3. Сетевые доступы 9
3. Задачи администратора 12
   3.1. Настройка параметров работы системы 12
      3.1.1. Установка агента 12
      3.1.2. Конфигурационный файл агента 19
      3.1.3. Конфигурация сервера 24
      3.1.4. Настройка Web-приложения 24
      3.1.5. Служба "saymon-server" 24
      3.1.6. Просмотр websocket-нотификаций 25
      3.1.7. Увеличение количества обработчиков SNMP-Trap 25
      3.1.8. Сброс системы к заводским настройкам 26
      3.1.9. Включение функционала генерации аварий 27
      3.1.10. Передача аварийных сообщений в сторонние сервисы 27
      3.1.11. Работа со свойствами аварий 28
      3.1.12. Перевод SNMP OID в текстовый формат 29
      3.1.13. Выполнение скриптов при создании и удалении объектов 30
      3.1.14. Автоматическое обнаружение объектов 31
   3.2. Модуль аналитики 33
      3.2.1. Запуск модуля аналитики в Центральном Пульте, установленном через ISO-образ 33
      3.2.2. Подключение модуля аналитики к Центральному Пульту в конфигурационных файлах 33
      3.2.3. Мониторинг работы модуля аналитики 34
   3.3. Управление сенсорами мониторинга 34
      3.3.1. Скрытие неиспользуемых сенсоров мониторинга 34
      3.3.2. Настройка пользовательских сенсоров 35
   3.4. Настройка уведомлений 37
      3.4.1. Настройка отправки SMS-уведомлений и голосовых вызовов 37
      3.4.2. Отправка почтовых уведомлений 38

3.4.3. Настройка уведомлений в Telegram 39
3.5. Интеграция со сторонними системами 41
3.5.1. Интеграция с Zabbix. 41
3.5.2. Интеграция с Grafana. 54
3.5.3. Интеграция с Keycloak. 58
3.6. Управление логированием 70
3.6.1. Конфигурация log-файлов агента. 70
3.6.2. Конфигурация log-файлов 72
3.6.3. Просмотр информации о Журнале событий 72
3.6.4. Назначение ответственного за событие 73
3.6.5. Установка ограничения для логирования 73
3.6.6. Удаление всех SNMP-Trap’ов из Журнала событий 74
3.6.7. Удаление логов. 76
3.7. Управление учётными записями пользователей. 76
3.7.1. Авторизация пользователей через LDAP. 76
3.7.2. Авторизация пользователей через Keycloak. 80
3.7.3. Создание учётной записи. 81
3.7.4. Назначение пользователям прав доступа. 82
3.7.5. Назначение парольных политик 84
3.7.6. Изменение пароля от учётной записи. 85
3.7.7. Настройка доступа к объектам. 86
3.7.8. Блокировка учётной записи 87
3.7.9. Удаление учётной записи. 88
3.8. Работа с объектами и связями 90
3.8.1. Создание объекта 91
3.8.2. Клонирование объекта 92
3.8.3. Удаление объекта 92
3.8.4. Создание ссылки на объект 93
3.8.5. Создание связи. 94
3.8.6. Удаление связи. 96
3.9. Настройка интерфейса 96
3.9.1. Выравнивание/расстановка объектов в стандартном виде. 96
3.9.2. Настройка заголовка web-интерфейса 97
3.9.3. Перемещение/отключение фоновой иконки объекта. 97
3.9.4. Вертикальное отображение имени объекта 97
3.9.5. Редактирование стилей состояний 98
3.10. Настройка мониторинга 99

3.10.1. Мониторинг основных параметров ПК. 99
3.10.2. Мониторинг процесса памяти 99
3.10.3. Мониторинг изменения файлов и папок 100
3.10.4. Проверка доступности web-ресурса 100
3.10.5. Безагентный мониторинг web-сервера 101
3.11. Страницы для внедрения в IFrame 102
3.11.1. Страница с авариями 102
3.11.2. Страница с графиками 104
3.11.3. Страница с журналом сессий 110
3.11.4. Страница журнала событий 111
3.12. Открытие терминала удалённого доступа. 117
3.13. Переход к web-интерфейсу устройства 118
3.14. Резервное копирование и восстановление данных 119
3.14.1. MongoDB 119
3.14.2. OpenTSDB. 121
3.14.3. Конфигурационные файлы 123
3.15. Расширения. 123
3.15.1. Установка расширений 123
3.15.2. Пользовательские расширения 124
4. Проблемы в работе системы и способы их решения 126
4.1. Недостаточно места на виртуальной машине с сервером 126
4.2. Отсутствие подключения агента к серверу 127
4.3. Проверка работы MongoDB 128
4.4. Проверка работы Redis. 128
4.5. 500 Internal Server Error и отсутствие графиков. 129
4.6. Ошибка работы HTTP-проверки 130
Приложение А (обязательное) 130
Приложение Б (обязательное) 136
Пример скрипта 139
Приложение В (обязательное) 142
Конфигурация сервера. 142
agent_stat 142
cache 143
custom_scripts 144
db 145
elasticsearch 145
encrypt 146

extensions . . . 147
hostname_update . . . 148
kafka . . . 148
keycloak . . . 150
ldap . . . 151
monitoring . . . 153
mqtt . . . 153
openTsdb . . . 154
push_notification . . . 155
resource_server . . . 156
rest_server . . . 157
server . . . 163
snmp . . . 172
state_history_log . . . 173
wetty . . . 173
zabbix . . . 174
Приложение Г (обязательное) . . . 177
Конфигурация Web-приложения . . . 177
agentInstallerUrl . . . 178
authEnabled . . . 179
authTokenEnabled . . . 179
authentication . . . 180
canEditObjectStyle . . . 181
comet . . . 182
configuration . . . 183
collapseSections . . . 183
defaultClassImage . . . 184
defaultEmbeddableLink . . . 185
defaultPieChartColors . . . 186
disabledSections . . . 186
documentation . . . 187
enableAnalytics . . . 190
enableJiraTrigger . . . 191
enableConditionalIncidents . . . 191
enableSmsTrigger . . . 192
enableSnmpTranslate . . . 192
enableVoiceCallTrigger . . . 192

forceEmptyStandardView 193
geoMap 194
grid 195
gridView 197
history 198
hideConnectionStatusNotifications 201
hideResponseErrorNotifications 202
incidents 202
languages 204
monitoring 206
millisecondMonitoring 206
pollInterval 207
pollIntervalSocket 207
statHistory 207
title 208
themes 208
tree 210
useNaturalSort 210

1. Введение

Центральный Пульт — это платформа для визуализации и мониторинга работы сети, оборудования, приложений и служб.

1.1. 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".

1.2. 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.

2. Системные требования

Система может быть установлена на выделенных аппаратных или виртуальных мощностях.

Для стабильной работы клиента системы необходим web-браузер Google Chrome версии не

ниже 58.0.

2.1. Системные требования к серверу

Для работы сервера системы требуется следующая конфигурация:

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

Гарантируется корректная работа инсталляционного скрипта сервера на следующих ОС:

- Ubuntu Linux 20.04;
- Ubuntu Linux 18.04.

Установка системы также возможна на операционные системы:

- РЕД ОС,
- Red Hat Enterprise Linux,
- Astra Linux,
- Debian.

"Центральный Пульт" может быть запущен и на других дистрибутивах Linux. Свяжитесь с нами welcome@saymon.info, если вы не нашли нужную вам операционную систему среди перечисленных.

Серверная часть системы может быть поставлена в виде готового образа виртуальной машины или Docker-контейнера. Ссылки можно получить в службе технической поддержки по адресу welcome@saymon.info.

2.2. Системные требования к агенту

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

- Ubuntu Linux;
- Red Hat Enterprise Linux / CentOS Linux;
- Raspberry Pi;
- macOS;
- 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

2.3. Сетевые доступы

В таблице ниже перечислены порты по умолчанию, необходимые для корректной работы Центрального Пульта:

Порт	Протокол	Сервис	Инициатор соединения	Место назначения	Комментарий
22	TCP	SSH	Сотрудники технической поддержки	Все серверы платформы	Удалённый доступ к серверам для настройки и оказания технической поддержки.
80	TCP	HTTP	Пользователи Центрального Пульта	Серверы с сервисами Центрального Пульта	Доступ к web-интерфейсу платформы. Рекомендуется перенаправление на 443 порт.
162	UDP	SNMP Trap	Оборудование, отправляющее SNMP Trap	Серверы с агентами	Получение SNMP Trap от оборудования. По умолчанию на каждом сервере с сервисами Центрального Пульта установлен агент, который получает SNMP Trap’ы.

Порт	Протокол	Сервис	Инициатор соединения	Место назначения	Комментарий
443	TCP	HTTPS	Пользователи Центрального Пульта	Серверы с сервисами Центрального Пульта	Доступ к web-интерфейсу Центрального Пульта с шифрованием трафика.
3001	TCP	Wetty	Пользователи Центрального Пульта	Оборудование в инфраструктуре заказчика	Доступ к терминалам устройств через SSH и Telnet.
4242	TCP	OpenTSDB	Серверы с сервисами Центрального Пульта	Серверы с OpenTSDB	Доступ к базе данных хранения временных рядов для построения графиков. Необходим при инсталляции OpenTSDB на отдельных серверах.
6379	TCP	Redis	Серверы с сервисами Центрального Пульта	Серверы с Redis	Доступ к кэшу данных. Необходим при инсталляции Redis на отдельных серверах.
8090	TCP	REST API	Пользователи, интеграционные модули и расширения	Серверы с сервисами Центрального Пульта	Взаимодействие с системой и данными через REST API.

Порт	Протокол	Сервис	Инициатор соединения	Место назначения	Комментарий
8091	TCP	Notification Actor	Пользователи Центрального Пульта	Серверы с сервисами Центрального Пульта	Необходим для быстрого обновления данных в web-интерфейсе и мобильных клиентах через WebSocket.
8092	TCP	Resource Server	Пользователи, интеграционные модули и расширения	Серверы с сервисами Центрального Пульта	Отображение в web-интерфейсе приложенных к объектам документов и фоновых изображений.
9092	TCP	Kafka	Серверы с сервисами Центрального Пульта и агентами	Серверы с Kafka	Доступ к шине данных. Необходим при инсталляции системы на нескольких серверах и при использовании агента для сбора данных.
27017	TCP	MongoDB	Серверы с сервисами Центрального Пульта	Серверы с MongoDB	Доступ к базе данных основной информации об объектах и пользователях. Необходим при инсталляции MongoDB на отдельных серверах.

3. Задачи администратора

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

3.1. 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 удалённых команд, выполняемых в случае возникновения проблем.

3.1.1. 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 на страницу "Загрузки" 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.

WARNING	*При установке агента на Linux в системе автоматически создаётся пользователь **saymon**, от имени которого будет работать агент.* *Если в системе уже есть пользователь с таким логином, работа агента будет проходить от его имени.*

3.1.1.1. Автоинсталляция

Для установки агента предусмотрена упрощённая процедура автоинсталляции, с помощью которой можно быстро установить агент на компьютер и связать с объектом на сервере.

NOTE	Подробнее об объектах можно прочитать в разделе Работа с объектами и связями.

Чтобы выполнить автоинсталляцию агента, необходимо проделать следующие шаги:

1. Открыть web-интерфейс платформы "Центральный Пульт".
2. Ввести логин и пароль учётной записи с правами на управление объектами, свойствами и документами.
3. Создать объект класса `Saymon Agent`.
4. Навести курсор на созданный объект и нажать на появившуюся иконку - **Настройки агента** (Рис. 1) :

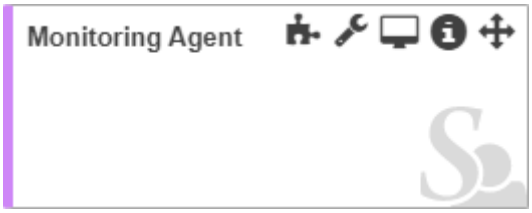


Рис. 1. Мониторинговый объект

5. В появившемся окне скопировать текст из поля `Команда для установки агента` (Рис. 2) :

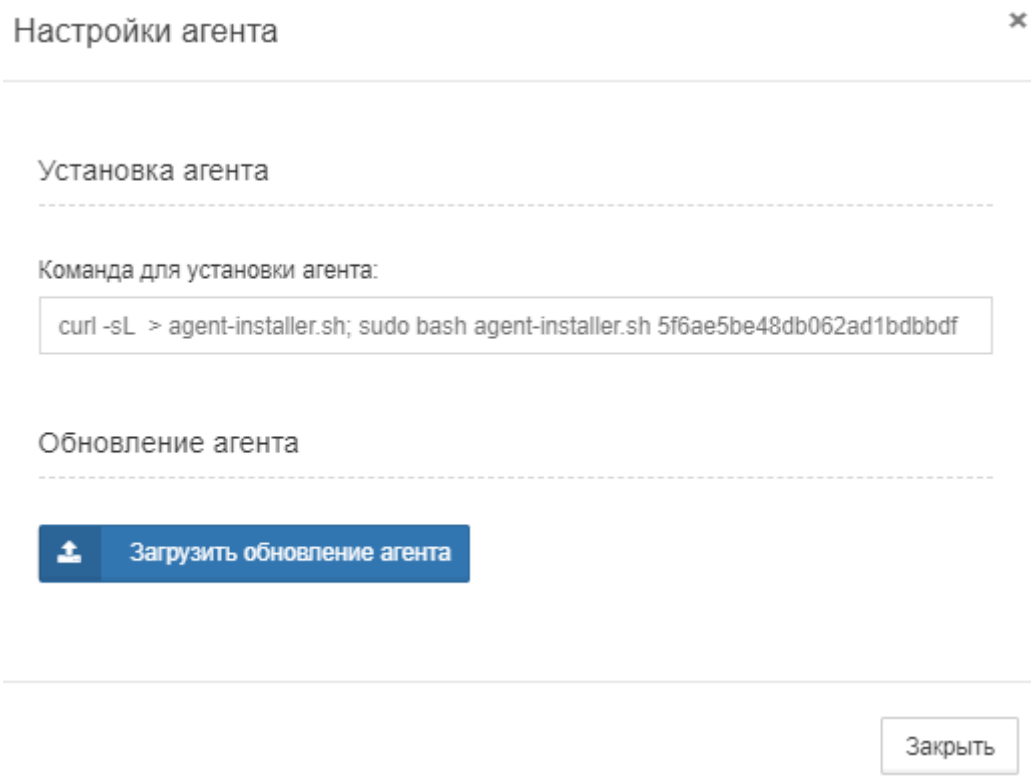


Рис. 2. Команда для установки агента

6. Выполнить данную команду на необходимом сервере.

NOTE	Последние 24 символа команды - уникальный ID агента.

В течение 30 секунд агент скачивается и устанавливается. Если установка произведена корректно, состояние объекта будет отражено зелёным цветом:



3.1.1.2. Linux

WARNING	При установке агента в системе автоматически создаётся пользователь **saymon**, от имени которого будет работать агент. Если в системе уже есть такой пользователь, то работа агента будет происходить от его имени.

Установка агента на хосте без доступа в Интернет

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

   64 bit

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

   32 bit

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

4. Запустить скрипт:

   64 bit

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

   32 bit

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

5. Выполнить дальнейшие инструкции в терминале.

Установка агента на хосте с доступом в Интернет

1. Выполнить однострочник:

   64 bit

   ```
   $ 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
   ```

32 bit

```
$ 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

Для изменения настроек нужно отредактировать конфигурационный файл агента `/opt/saymon-agent/conf/agent.properties`.

Поддерживаемые версии Linux

Агент работает со следующими версиями ОС Linux:

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

3.1.1.3. macOS

Для установки агента на macOS необходимо:

1. Создать системного пользователя **Saymon agent**, из-под которого будет запускаться агент.
2. Скачать архив `Mac OS X 64 bit` и распаковать его в папку `/opt`.
3. Отредактировать конфигурационный файл 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. Сделать файл агента исполняемым:

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

6. Запустить агента:

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

3.1.1.4. Windows

Установка

Для установки агента на Windows необходимо:

1. Скачать инсталлятор для ОС `Windows 64-bit` или `Windows 32-bit`.
2. Запустить скачанный файл от имени администратора.
3. При необходимости изменить каталог установки по умолчанию (для `Windows 64-bit` - `C:\Program Files (x86)\SAYMON Agent`, для `Windows 32-bit` - `C:\Program Files\SAYMON Agent`) и нажать кнопку Install (Рис. 3) .

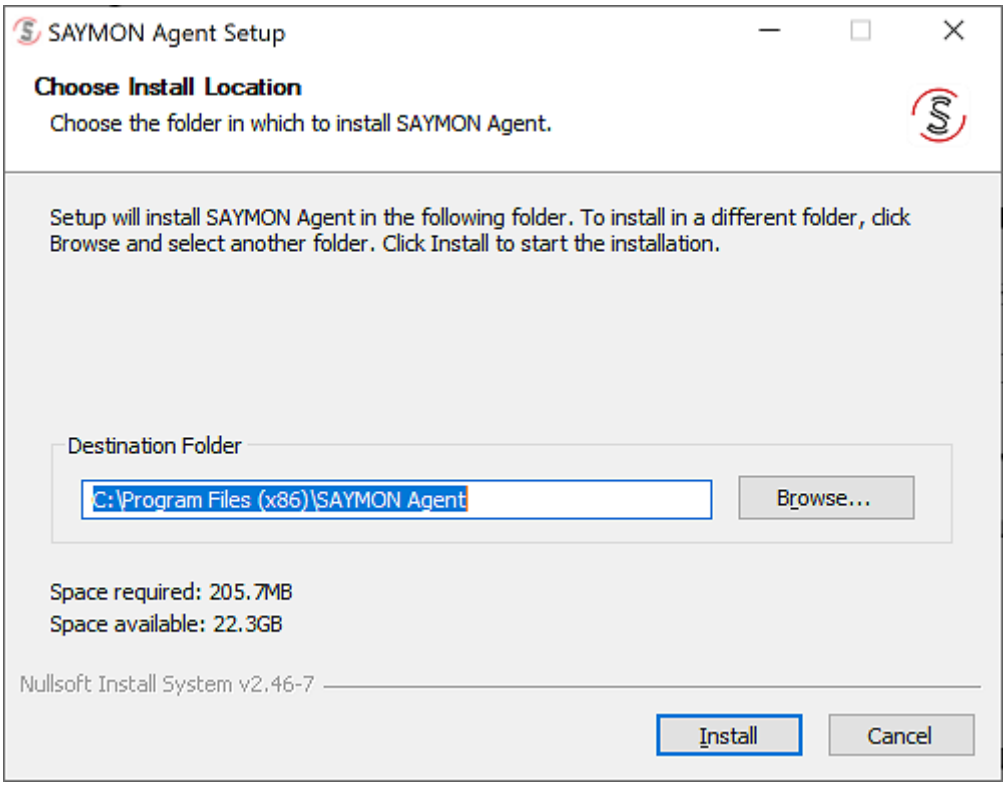


Рис. 3. Выбор каталога установки

4. После завершения установки нажать кнопку Close (Рис. 4) .

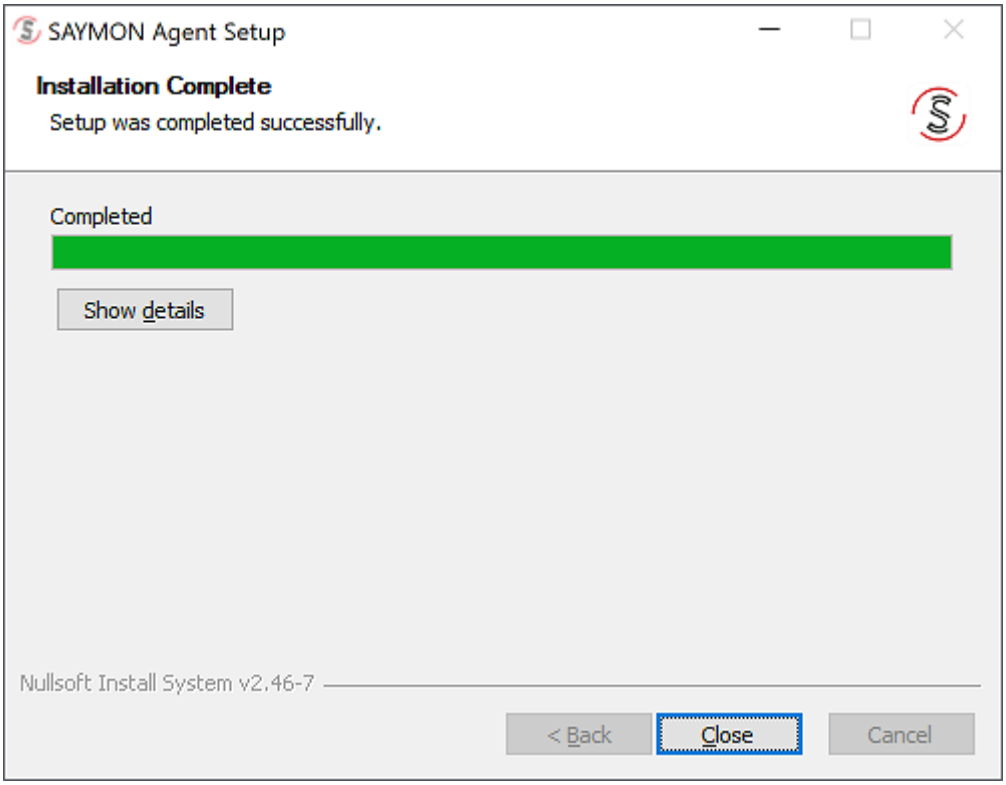


Рис. 4. Завершение установки

5. В открывшемся окне ознакомиться с дальнейшими инструкциями по установке агента и нажать кнопку OK (Рис. 5) .

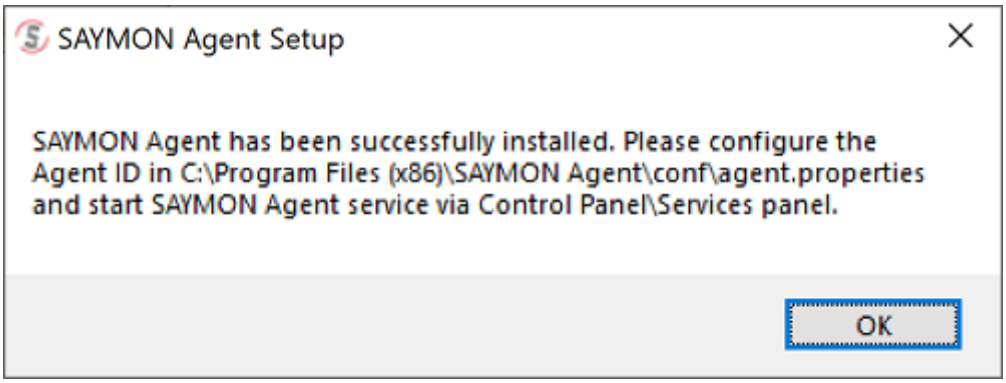


Рис. 5. Первичная настройка агента

6. Перейти в папку `conf` каталога установки агента и открыть файл **agent.properties** (Конфигурационный файл агента).
7. В качестве значения параметра **server.host** указать IP-адрес сервера Центрального Пульта, к которому должен подключиться агент. Закрыть файл, сохранив изменения.
8. Открыть оснастку **Службы** и запустить службу с именем **saymon-agent**.
9. После регистрации агента на сервере создается новый объект класса Saymon Agent. Идентификатор созданного объекта (**agentId**) передается агенту. На хосте с агентом создается файл **registration.json** (для `Windows 64-bit` - `C:\Program Files (x86)\SAYMON Agent\storage`, для `Windows 32-bit` - `C:\Program Files\SAYMON Agent\storage`), в котором указан данный идентификатор.
10. Чтобы инициировать процедуру получения **agentId** повторно (например, если нужно

подключить агента к другому серверу), достаточно удалить файл **registration.json** и перезапустить службу агента.

Поддерживаемые версии ОС

Агент работает со следующими версиями ОС Windows:

- Windows 7 SP1, 64-bit;
- Windows 7 SP1, 32-bit;
- Windows 10 22H2, 64-bit;
- Windows 10 22H2, 32-bit;
- Windows 10 Enterprise, 64-bit;
- Windows 10 Enterprise, 32-bit;
- Windows 11 Enterprise 22H2, 64-bit;
- Windows Server 2012 (R2 Datacenter), 64-bit;
- Windows Server 2016 (R2 Datacenter), 64-bit;
- Windows Server 2019, 64-bit;
- Windows Server 2022 (Standard Evaluation 21H2), 64-bit.

3.1.1.5. Wiren Board 6

Wiren Board 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".

Инструкция описывает пошаговый процесс установки и настройки агента на контроллер автоматизации Wiren Board 6.

Благодарим за оказанную помощь пользователя svdu с форума компании Wiren Board.

1. Скачать архив с агентом.
2. Распаковать архив в папку `/opt/saymon-agent/`.
3. Скачать архив с JDK (потребуется регистрация 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
```

3.1.2. Конфигурационный файл агента

WARNING	Для применения изменений необходимо перезапустить службу • **saymon-agent** (Linux/Mac OS); • **SaymonAgent** (Windows).

Далее описаны параметры конфигурационного файл агента `<Папка_установки_агента>/conf/agent.properties`.

3.1.2.1. Общие настройки

Параметр	Описание
agent.discoveryEnabled	Включает автоматический поиск агентом сетевых устройств. • false - автоматический поиск выключен, • true - автоматический поиск включен. По умолчанию - **false** - автоматический поиск выключен. **NOTE** Задать родительский объект для обнаруженных устройств можно с помощью параметра `discovery_parent_id` в разделе **Server** конфигурационного файла сервера.

Параметр	Описание
agent.id	Уникальный идентификатор объекта класса `Saymon Agent` в web-интерфейсе системы. **NOTE** При указании agent.id=0 на сервере будет создан новый объект класса `Saymon Agent`. По умолчанию агенты создаются в корневом объекте, задать родительский объект для создаваемых агентов можно с помощью параметра `agent_registration_parent_id` в разделе **Server** конфигурационного файла сервера. ID созданного объекта будет передан агенту. На хосте с агентом будет создан файл `.../saymon-agent/storage/registration.json`, где будет записан данный идентификатор. Чтобы заново инициировать процедуру получения ID, к примеру, если необходимо подключить агента к другому серверу, достаточно удалить файл `registration.json` и перезапустить агента.

Параметр	Описание
agent.scriptsEnabled	Включает выполнение агентом скриптов с указанным текстом; • false - выполнение скриптов выключено, • true - выполнение скриптов включено. По умолчанию - **true** - выполнение скриптов включено.
agent.snmpTrapEnabled	Позволяет использовать агента в качестве получателя трапов. • false - получение трапов агентом выключено, • true - получение трапов агентом включено. По умолчанию - **false** - получение трапов агентом выключено.
agent.snmpTrapListenPort	Порт для получения трапов. По умолчанию - **1162**.
agent.snmpTrapReceiverThreadPoolSize	Количество одновременных потоков для получения трапов. По умолчанию - **4**.

Параметр	Описание
metrics.buffer.in-memory.capacity	Включает буфер для сохранения данных от агента при отсутствии связи между агентом и сервером. Значение параметра - количество наборов данных от агента (результатов проверок), которые будут храниться в буфере до восстановления связи с сервером. После восстановления связи все данные из буфера пересылаются на сервер, и буфер очищается. При переполнении буфера очередные наборы данных заменяют наиболее старые из имеющихся в буфере. По умолчанию буфер выключен. **WARNING** Функционал поддерживается агентами версии 4.5.72 и выше.
server.host	Адрес сервера Kafka. По умолчанию - **127.0.0.1**.
server.port	Порт Kafka, по которому осуществляется подключение агентов к серверу. По умолчанию - **9092**.

3.1.2.2. Подключение к Kafka Cluster

При использовании Kafka Cluster необходимо также использовать дополнительные параметры подключения, начинающиеся на `kafka.`; все возможные параметры представлены в документации по Kafka.

Пример:

```
...
kafka.bootstrap.servers=192.168.1.11:9092,192.168.1.12:9092,192.168.1.13:9092
kafka.security.protocol=SASL_SSL
kafka.ssl.truststore.location=/opt/kafka/kafka.client.truststore.jks
kafka.ssl.truststore.password=saymon
```

```
kafka.sasl.mechanism=PLAIN
kafka.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="user" password="password"
...
```

Параметр	Описание
kafka.bootstrap.servers	Список пар хост:порт, использующихся для установки первоначального соединения с кластером Kafka.

WARNING	При использовании Kafka Cluster поле **server.host** не должно быть пустым, там необходимо указать адрес одного из **kafka.bootstrap.servers**.

3.1.2.3. Авторизация и шифрование

Параметр	Описание
kafka.sasl.jaas.config	Параметры контекста входа в систему для SASL-соединений в формате, используемом конфигурационными файлами JAAS.
kafka.sasl.mechanism	Механизм SASL, используемый для клиентских подключений.
kafka.security.protocol	Протокол, используемый для взаимодействия с брокерами. Возможные значения: • PLAINTEXT, • SSL, • SASL_PLAINTEXT, • SASL_SSL.
kafka.ssl.truststore.location	Расположение файла **truststore**.
kafka.ssl.truststore.password	Пароль для файла **truststore**. Если пароль не задан, сконфигурированный файл также будет использоваться, но без проверки целостности.

3.1.3. 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
```

TIP	Описание всех параметров конфигурации сервера доступно в Приложении В.

Управление некоторыми настройками доступно в web-интерфейсе системы:

1. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
2. Перейти в раздел "Опции конфигурации".

Изменённые параметры в интерфейсе приоритетнее настроек в конфигурационном файле. Установленные значения параметров автоматически сохраняются в MongoDB, конфигурационный файл не перезаписывается.

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
```

3.1.4. Настройка Web-приложения

Некоторые элементы web-интерфейса можно настроить в конфигурационном файле клиента `/etc/saymon/saymon-client.yaml`.

TIP	Описание всех параметров конфигурации сервера доступно в Приложении Г.

3.1.5. 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
```

3.1.6. 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 аварией и выбрать пункт "Подтвердить".

В консоли отобразится websocket-нотификация.

3.1.7. 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 число 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
```

3.1.8. Сброс системы к заводским настройкам

Администратор 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.
```

```
# Let's 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*
```

3.1.9. Включение функционала генерации аварий

Функционал генерации аварий опционален и выключен по умолчанию. Для его активации необходимо:

1. В секции **Server** конфигурационного файла сервера **/etc/saymon/saymon-server.conf** установить параметру **conditional_incidents_enabled** значение **true**:

```
{
    ...
    "server" : {
        ...
        "conditional_incidents_enabled ": true,
    },
...
}
```

2. В конфигурационном файле клиента **/etc/saymon/saymon-client.yaml** установить параметру **enableConditionalIncidents** значение **true**:

```
enableConditionalIncidents: true
```

3.1.10. Передача аварийных сообщений в сторонние сервисы

Для передачи аварийных сообщений в системы верхнего уровня необходимо выполнить следующие действия:

1. Включить функционал генерации аварий в разделе **Server** конфигурационного файла сервера `/etc/saymon/saymon-server.conf` и там же указать путь к скрипту,

который будет отправлять данные:

```
"server" : {
    "conditional_incidents_enabled" : true,
    "conditional_incidents_script":
"/home/saymon/scripts/send_incident.sh",
    ...
}
```

2. Добавить в MongoDB триггеры по авариям для классов или отдельных объектов:

```
mongo saymon
db.incidentTriggers.insert({ "ownerId" : 1042, "ownerType" : 4, "body" : [
{ "type" : "conditional_incidents_script", "payload" : { }, "states" : [
1,2,4] } ] })
```

где:

- ownerId - идентификатор сущности, к которой добавляется триггер;
- ownerType - тип сущности, для которой добавляется триггер (4 - класс, 1 - объект);
- states - уровень критичности аварий, при которых запускается триггер.

В указанном выше примере скрипт триггера будет вызываться при возникновении аварии любой критичности у объектов класса **1042.**

В скрипт передаётся два аргумента:

- $1 - тело аварии,
- $2 - свойства объекта, к которому относится авария.

3.1.11. Работа со свойствами аварий

Добавить/изменить или удалить свойства аварий можно с помощью REST API.

При желании можно добавлять в списки активных и исторических аварий дополнительные поля из свойств аварий, значения которых будут отражены в списках. Для этого необходимо указать нужные свойства аварий в разделе **Incidents** конфигурационного файла клиента **/etc/saymon/saymon-client.yaml**.

```
incidents:
  additionalFields:
    - properties.address,
  additionalFieldNames:
    - IP-address
```

REST API метод для добавления или изменения свойств аварии:

`POST /node/api/incidents/:id/props`

Пример (bash) добавления аварии свойств **foo** со значением **1** и **bar** со значением **second**:

```
login=<...>
password=<...>
saymon_hostname=<...>
incident_id=<...>
url=https://$saymon_hostname/node/api/incidents/$incident_id/props

curl -X POST $url -u $login:$password \
-H "Content-Type: application/json" \
-d @- <<EOF
{
"foo": 1,
"bar": "second"
}
EOF
```

REST API метод для удаления свойств аварии:

`DELETE /node/api/incidents/:id/props`

Пример (bash) удаления свойства **foo** аварии:

```
login=<...>
password=<...>
saymon_hostname=<...>
incident_id=<...>
url=https://$saymon_hostname/node/api/incidents/$incident_id/props

curl -X DELETE $url -u $login:$password \
-H "Content-Type: application/json" \
-d @- <<EOF
{
"foo"
}
EOF
```

WARNING	*Пустой запрос удаляет все свойства аварии.*

3.1.12. Перевод SNMP OID в текстовый формат

По умолчанию в результатах проверки SNMP Trap OID отображается в числовом формате.

Для перевода OID в "человекочитаемый" текстовый формат необходимо:

1. В секции **rest_server** конфигурационного файла сервера **/etc/saymon/saymon-server.conf** задать адрес сервера для трансляции OID в текстовый формат в параметре **snmp_mib_url**:

```
{
    ...
    "rest_server" : {
        ...
        "snmp_mib_url": "http://192.168.1.189:5550",
    },
    ...
}
```

2. В конфигурационном файле клиента **/etc/saymon/saymon-client.yaml** установить параметру **enableSnmpTranslate** значение **true**:

```
*enableSnmpTranslate: true*
```

3.1.13. Выполнение скриптов при создании и удалении объектов

При создании или удалении объектов того или иного класса система способна выполнять скрипты и передавать им аргументы, настраиваемые через web-интерфейс.

Каждый такой скрипт состоит из двух файлов:

- исполняемый файл;
- файл конфигурации, описывающий web-интерфейс вкладки Триггеры для передачи аргументов исполняемому файлу.

Файл конфигурации должен:

- находиться в одной папке с исполняемым файлом;
- иметь название исполняемого файла с дополнительным расширением *.conf*:

```
/opt/saymon-extensions/entity-triggers-path/myscript.sh
/opt/saymon-extensions/entity-triggers-path/myscript.sh.conf
```

Исполняемые файлы скриптов можно группировать в подкаталогах с любым уровнем вложенности:

```
/opt/saymon-extensions/entity-triggers-path/for_host/script_1.sh
/opt/saymon-extensions/entity-triggers-path/for_host/script_1.sh.conf
/opt/saymon-extensions/entity-triggers-path/for_memory/script_1.sh
/opt/saymon-extensions/entity-triggers-path/for_memory/script_1.sh.conf
/opt/saymon-extensions/entity-triggers-path/for_router/script_1.sh
```

```
/opt/saymon-extensions/entity-triggers-path/for_router/script_1.sh.conf
```

При добавлении или модификации скриптов все изменения применяются автоматически - достаточно обновить страницу в браузере.

3.1.13.1. Каталог для скриптов

Каталог скриптов задается в конфигурационном файле сервера `/etc/saymon/saymon-server.conf`.

Необходимо:

1. Добавить в файл секцию **custom_scripts.**
2. В параметре **entity_triggers_path** задать полный путь до папки со скриптами.

```
...
"custom_scripts": {
    "entity_triggers_path": "/opt/saymon-extensions/entity-triggers-path"
},
...
```

3. Перезапустить сервер:

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

NOTE	Возможные поля файла настроек параметров и пример конфигурационного файла представлены в Приложении Б.

3.1.14. Автоматическое обнаружение объектов

Агент сканирует локальную подсеть и отправляет на сервер информацию обо всех IP-адресах, доступных для TCP-соединения либо доступных по PING. Для обнаруженных хостов объекты класса **Host** с соответствующими адресами создаются в Центральном Пульте и отображаются на карте инфраструктуры. Внутри каждого из этих объектов автоматически создаётся объект класса **Ping**, в котором включается мониторинг с помощью агента, выполнившего автообнаружение, и сенсора Ping.

IP-адрес обнаруженного объекта автоматически записывается в свойство **IP**, что можно использовать для перехода по этому адресу в браузере или доступа к объекту по SSH или Telnet через контекстное меню.

Чтобы настроить автообнаружение объектов, необходимо:

1. В конфигурационном файле агента **.../saymon-agent/conf/agent.properties** установить параметру **agent.discoveryEnabled** значение **true**:

```
*agent.discoveryEnabled=true*
```

2. Указать ID родительского объект для обнаруживаемых объектов в параметре **server.discovery_parent_id** конфигурационного файла сервера **/etc/saymon/saymon-server.conf**:

WARNING	*По умолчанию обнаруженные объекты помещаются в корневой объект.*

```
...
"server": {
    ...
    "discovery_parent_id": "632aae8b1a687b43378c0657",
    ...
},
...
```

3. Перезапустить службы saymon-agent и saymon-server:

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

3.2. Модуль аналитики

3.2.1. Запуск модуля аналитики в Центральном Пульте, установленном через ISO-образ

По умолчанию модуль аналитики, используемый для прогнозирования и определения всплесков, не включен в ISO-образ Центрального Пульта. Чтобы установить модуль аналитики, необходимо:

1. Скачать скрипт analytics-create-and-run.sh и разместить его в файловой системе на виртуальной машине, где установлен ISO-образ Центрального Пульта.
2. Запустить скрипт командой

```
$ sudo ANALYTICS_BRANCH=x.y.z ./analytics-create-and-run.sh
```

WARNING	x.y.z - версия модуля аналитики, совместимая с вашей версией Центрального Пульта.

NOTE	Проверить, что контейнер **saymon-analytics** запустился и работает, можно с помощью команды `$ docker ps -a`

3.2.2. Подключение модуля аналитики к Центральному Пульту в конфигурационных файлах

Чтобы подключить модуль аналитики, потребуется внести изменения в конфигурацию Центрального Пульта:

1. В раздел **Server** конфигурационного файла сервера `/etc/saymon/saymon-server.conf` добавить следующие параметры:

```
  "server": {
    "analytics_enabled": true, # включает аналитику метрик в процесс
обработки данных в системе
    "analytics_processes": 2 # количество логических ядер, выделяемых для
обработки аналитики (по умолчанию используются все доступные ядра системы),
    ...
  }
```

2. В конфигурационном файле клиента `/etc/saymon/saymon-client.yaml` установить параметру **enableAnalytics** значение **true**:

```
enableAnalytics: true
```

3. Перезапустить сервер Центрального Пульта:

```
$ service saymon-server restart
```

3.2.3. Мониторинг работы модуля аналитики

Чтобы мониторить работу модуля аналитики, можно использовать логи контейнера **saymon-analytics**. Для этого необходимо:

1. Подключиться к контейнеру командой

```
$ docker exec -ti saymon-analytics bash
```

2. Открыть файл логов модуля командой

```
$ tail -f /opt/analytics/analytics.log
```

В логах модуля содержится информация о

- настройках, с которыми запущен модуль (слушающий сокет и количество задействованных в пуле обработчика процессов),
- обрабатываемых метриках,
- результатах обработки метрик,
- ошибках, возникающих в процессе работы модуля.

3.3. Управление сенсорами мониторинга

Системой предусмотрена возможность скрывать в интерфейсе неиспользуемые мониторинговые сенсоры, а также настраивать пользовательские сенсоры и внедрять их в интерфейс.

3.3.1. Скрытие неиспользуемых сенсоров мониторинга

Системой предусмотрена возможность скрывать в web-интерфейсе неиспользуемые сенсоры мониторинга.

Для скрытия сенсоров необходимо:

- добавить в конфигурационный файл клиента `/etc/saymon/saymon-client.yaml` раздел **Monitoring**;
- в переменной **monitoring.standardTasks** указать массив идентификаторов отображаемых сенсоров.

```
monitoring:
  standardTasks:
    - snmpGet
    - snmpTrap
    - ping
```

CAUTION	Идентификаторы сенсоров чувствительны к регистру символов.

3.3.1.1. Список идентификаторов стандартных сенсоров

В системе есть следующие стандартные сенсоры:

- **binaryProtocol** - бинарный протокол,
- **configFile** - конфигурационный файл / директория,
- **databaseQuery** - запрос в базу данных,
- **exec** - выполнение программы / скрипта,
- **external** - внешняя проверка (записанные через API данные),
- **ftp** - FTP-сенсор,
- **httpRequest** - HTTP-запрос,
- **jmx** - JMX-сенсор,
- **localNetworkPort** - локальный порт,
- **mqtt** - MQTT-сенсор,
- **ping** - пинг-сенсор,
- **processInfo** - процесс по имени,
- **remoteNetworkPort** - удалённый порт,
- **snmpGet** - SNMP_Get-сенсор,
- **snmpTrap** - SNMP_Trap-сенсор,
- **wmi** - WWMI-сенсор.

3.3.2. Настройка пользовательских сенсоров

В качестве сенсора можно использовать собственноручно написанный скрипт и передавать ему в качестве аргументов значения параметров через удобный web-интерфейс.

Для применения изменений достаточно обновить страницу в браузере.

Чтобы сделать пользовательский скрипт мониторинговым сенсором, необходимо:

1. Задать каталог для хранения пользовательских сенсоров:

a. Открыть конфигурационный файл сервера **/etc/saymon/saymon-server.conf** и добавить в него секцию **monitoring**:

```
{
    ...
    "monitoring": {
        "custom_tasks_path": "/opt/saymon-agent/custom_tasks"
    },
    ...
}
```

2. В параметре **monitoring.custom_tasks_path** указать полный путь до каталога.
3. Поместить исполняемый файл в указанный каталог.

WARNING	*Исполняемые файлы пользовательских сенсоров могут располагаться в любом количестве подпапок. Поиск производится рекурсивно.*

4. Проверить идентификатор пользовательского сенсора. Идентификатор сенсора создаётся автоматически и эквивалентен имени исполняемого файла с относительным путём (если есть). Например: **utils/network/ping.**

WARNING	*Идентификатор пользовательского сенсора не должен совпадать с идентификатором стандартного, иначе пользовательский сенсор будет проигнорирован.*

5. Задать настройки сенсора в конфигурационном файле.

   Параметры сенсора определяются в JSON-файле, именуемом как **имя_исполняемого_файла + .conf**. Располагаться этот файл должен в том же каталоге, что и исполняемый файл.

   Пример:

```
/opt/saymon-agent/custom_tasks/mytask.sh

/opt/saymon-agent/custom_tasks/mytask.sh.conf
```

NOTE	*Возможные поля файла настроек параметров и пример конфигурационного файла представлены в Приложении Б.*

WARNING	*Пользовательские сенсоры выполняются агентами локально, поэтому необходимо скопировать исполняемые файлы в одну и ту же папку на все хосты, на которых предполагается их запускать.*

	Конфигурационные файлы достаточно разместить на хостах с сервером системы.

3.4. Настройка уведомлений

При переходе объектов в определенные состояния система может:

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

Уведомления настраиваются в виде подробной информации об объекте в секции "Действия при смене состояния" (Рис. 6):

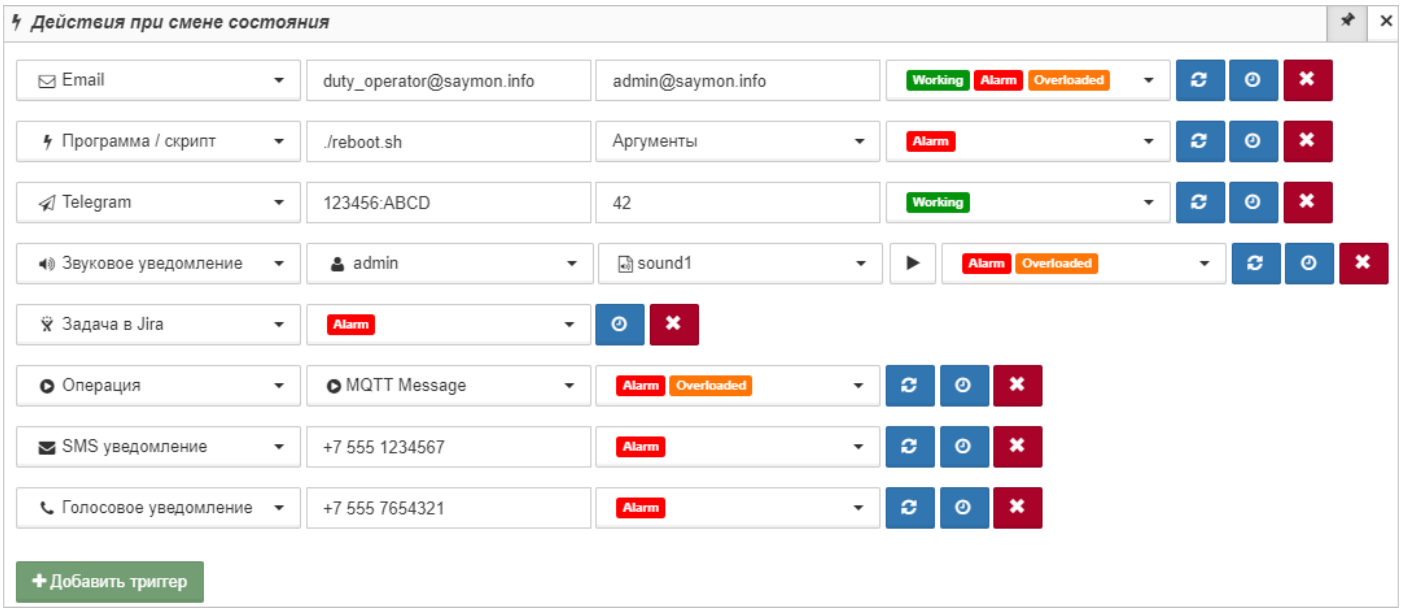


Рис. 6. Действия при смене состояния

3.4.1. Настройка отправки SMS-уведомлений и голосовых вызовов

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

1. В файле конфигурации клиента `/etc/saymon/saymon-client.yaml` установить параметрам `enableSmsTrigger` и `enableVoiceCallTrigger` значения `true`:

```
enableVoiceCallTrigger : true
enableSmsTrigger : true
```

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

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

WARNING	Скрипт, отправляющий SMS-уведомления, получает в качестве аргументов: $1 - номер телефона получателя (как введён 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.

WARNING	Скрипт, осуществляющий голосовой вызов, получает в качестве аргументов: $1 - номер телефона получателя (как введён в триггере); $2 - ID объекта, в котором сработал триггер; $3 - имя объекта, в котором сработал триггер; $4 - ID состояния объекта.

3. Чтобы применить изменения, необходимо перезапустить сервер:

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

3.4.2. Отправка почтовых уведомлений

Для настройки отправки почтовых уведомлений требуется настроить необходимые

параметры доступа к почтовому серверу в подсекции **Email** секции **Server** файла **/etc/saymon/saymon-server.conf**.

Пример настройки подсекции **Email**:

```
"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
    },
}
```

3.4.3. Настройка уведомлений в Telegram

Для настройки отправки уведомлений с помощью Telegram необходимо:

1. Задать бота, от которого будут поступать уведомления:
    a. Создать нового бота:
        i. Найти в приложении Telegram контакт **BotFather**.
        ii. Отправить ему сообщение **/newbot**.
        iii. Задать боту отображаемое имя (**name**, позже возможно изменить).
        iv. Задать боту уникальное имя (**username,** изменить будет невозможно).
        v. Скопировать токен бота вида `210979209:AAFfT2mt3oW4EK1gYqE_d3OjAJSIRLSrAL`.
        vi. Отправить контакту **BotFather** сообщение **/setprivacy**.
        vii. Выбрать созданного бота по его **username**.
        viii. Выбрать опцию **Disable**.
    b. Использовать существующего бота:
        i. Найти в приложении Telegram контакт **BotFather**.
        ii. Отправить ему сообщение **/mybots**.
        iii. Выбрать нужного бота.

iv. Выбрать опцию **API Token**.

v. Скопировать токен бота вида `210979209:AAFfT2mt3oW4EK1gYqE_d3OjAJSIRLSrAL`.

2. Настроить канал, чат или группу:

a. Создать приватный канал (рекомендуется):

i. В приложении Telegram создать новый канал.

ii. Открыть настройки канала, добавить бота в список администраторов.

iii. Отправить сообщение в канал.

iv. 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
```

v. Найти текст со словами **chat** и **id**, например

```
..."channel_post":{"message_id":4,"chat":{"id":-1001156346945
,"title":"SAYMON"... }}
```

Здесь **-1001156346945** - искомый ID канала.

b. Создать чат:

i. Отправить боту любое сообщение.

ii. 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
```

iii. Найти текст со словами **chat** и **id**, например

```
... ":"K"},"chat":\{"id":121399918,"first_ ...
```

Здесь **121399918** - искомый ID чата.

c. Настроить группу:

i. Добавить бота в группу.

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

iii. 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
```

iv. Найти текст со словами **chat** и **id**, например

```
... ":"K"},"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.

3.5. Интеграция со сторонними системами

Система предусматривает возможность импортировать данные с других мониторинговых платформ и экспортировать метрики в платформы для визуализации данных.

3.5.1. Интеграция с Zabbix

Система позволяет подключаться к нескольким серверам Zabbix и импортировать информацию хостах, которые находятся на мониторинге.

3.5.1.1. Настройка подключения к Zabbix 5.x

Для настройки подключения нужно выполнить следующие шаги:

Шаг 1: проверить доступ от сервера системы до сервера Zabbix

Для проверки доступности Zabbix API необходимо выполнить следующую команду в терминале на сервере системы, заменив `http://192.168.1.215` на адрес файла `api_jsonrpc.php` вашей инсталляции Zabbix:

```
$ curl -H "Content-Type: application/json" -X POST -d
'{"jsonrpc":"2.0","method":"apiinfo.version","id":1,"auth":null,"params":{}}'
http://192.168.1.215/zabbix/api_jsonrpc.php
```

При наличии доступа сервер Zabbix ответит похожим сообщением:

```
{"jsonrpc":"2.0","result":"3.2.1","id":1}
```

Шаг 2: добавить пользователя в Zabbix

Добавить в Zabbix пользователя с правами на чтение хостов или групп хостов, данные с которых необходимо импортировать. Имя и пароль данного пользователя будут использоваться для импорта данных на шаге 4.

Шаг 3: создать объект для импорта данных

На сервере системы нужно создать объект, в который будут импортироваться данные о хостах из Zabbix, и запомнить его ID, указанный в секции параметров, для использования на шаге 4.

Шаг 4: включить импорт данных на сервере системы

Добавить раздел **Zabbix** в конфигурационный файл сервера `/etc/saymon/saymon-server.conf` и внести в него следующую информацию:

```
"zabbix" : [{
    "url" : "http://192.168.1.215/zabbix/api_jsonrpc.php",
    "user" : "saymon",
    "password" : "saymon_user_password",
    "parent_id" : "58b586d5c3a2f96642e25537",
    "zabbix5": true
}]
```

где

- **url** – адрес файла `api_jsonrpc.php` вашей инсталляции Zabbix, который был использован на шаге 1, например: `http://192.168.1.215/zabbix/api_jsonrpc.php`;
- **user** – имя пользователя, добавленного в Zabbix на шаге 2;
- **password** – пароль пользователя, добавленного в Zabbix на шаге 2;
- **parent_id** – id объекта, созданного на шаге 3;
- **zabbix5** – включить поддержку Zabbix 5.x.

Для каждого дополнительного сервера Zabbix нужно добавить новый JSON-документ с его данными в массив следующим образом:

```
"zabbix" : [
    {
        "url" : "http://192.168.1.215/zabbix/api_jsonrpc.php",
        "user" : "saymon",
        "password" : "saymon_user_password",
        "parent_id" : "58b586d5c3a2f96642e12345",
        "zabbix5": true
    },
    {
        "url" : "http://192.168.1.180:8020/zabbix/api_jsonrpc.php",
        "user" : "saymon",
        "password" : "saymon_user_password",
        "parent_id" : "58b586d5c3a2f966427abcde",
        "zabbix5": true
```

```
    }
]
```

Шаг 5: применить настройки

Для применения настроек подключения к серверу Zabbix необходимо перезапустить сервер Центрального Пульта:

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

3.5.1.2. Настройка подключения к Zabbix 6.0

WARNING	Интеграцию с Zabbix 6.0 поддерживает только сервер Центрального пульта, начиная с версии 3.14.89. Более ранние версии сервера поддерживают только интеграцию с Zabbix 5.x.

NOTE	Ниже приводится описание интеграции с Zabbix 6.0, запущенным из docker-контейнера.

Чтобы настроить интеграцию Центрального пульта с Zabbix 6.0, выполните следующие действия:

Шаг 1: запуск Zabbix

1. Склонируйте репозиторий Zabbix:

   ```
   $ git clone https://github.com/zabbix/zabbix-docker
   ```

2. Выберите нужный docker-файл из склонированного репозитория и запустите его, например:

   ```
   $ docker-compose -f ./docker-compose_v3_alpine_mysql_latest.yaml up -d
   ```

Шаг 2: настройка автоматического запуска Zabbix

Чтобы при перезапуске сервера контейнеры запускались автоматически, создайте **systemd-unit**, который будет запускать **docker-compose** файл при загрузке системы. Для этого создайте файл `/etc/systemd/system/zabbix-compose.service` и вставьте в него следующее содержимое:

```
[Unit]
Description=Zabbix services with docker-compose
Requires=docker.service
After=docker.service
```

```
[Service]
WorkingDirectory=/opt/zabbix-docker/
User=root
Group=docker
Type=oneshot
RemainAfterExit=yes

ExecStartPre=/usr/local/bin/docker-compose -f ./docker-
compose_v3_alpine_mysql_latest.yaml down -v

# Compose up
ExecStart=/usr/local/bin/docker-compose -f ./docker-
compose_v3_alpine_mysql_latest.yaml up -d

# Compose down, remove containers
ExecStop=/usr/local/bin/docker-compose -f ./docker-
compose_v3_alpine_mysql_latest.yaml down

[Install]
WantedBy=multi-user.target
```

Для применения изменений последовательно выполните следующие команды:

```
$ systemctl enable zabbix-compose.service

$ systemctl daemon-reload

$ systemctl start zabbix-compose.service
```

Шаг 3: загрузка и запуск Zabbix Agent

Скачайте и установите Zabbix Agent.

Для работы с сервисом Zabbix Agent используйте указанные ниже команды.

Редактирование конфигурационного файла:

```
$ sudo nano /etc/zabbix/zabbix_agentd.conf
```

Запуск сервиса:

```
$ sudo systemctl start zabbix-agent
```

Проверка состояния сервиса:

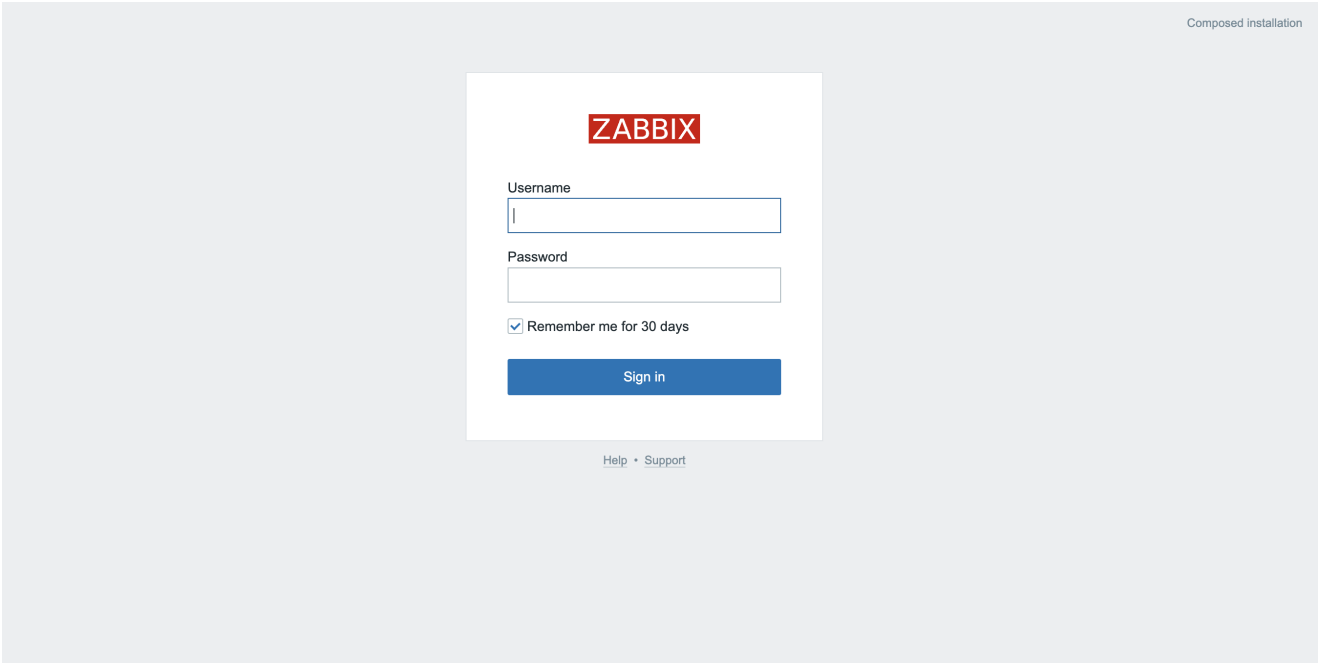
```
$ sudo systemctl status zabbix-agent
```

Остановка сервиса:

```
$ sudo systemctl stop zabbix-agent
```

Шаг 4: подключение Zabbix Agent

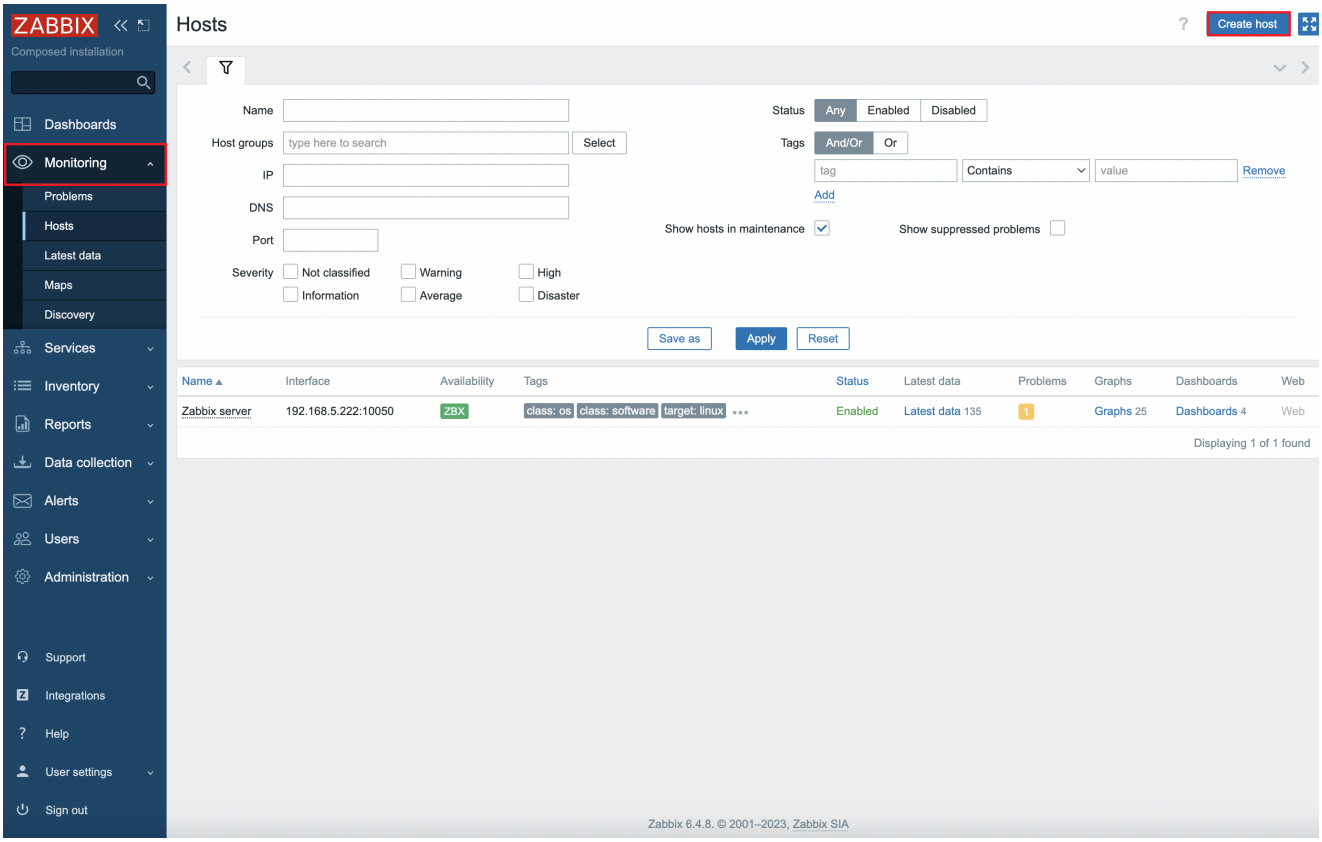
1. Выполните вход в web-интерфейс сервера Zabbix (http://host/zabbix).



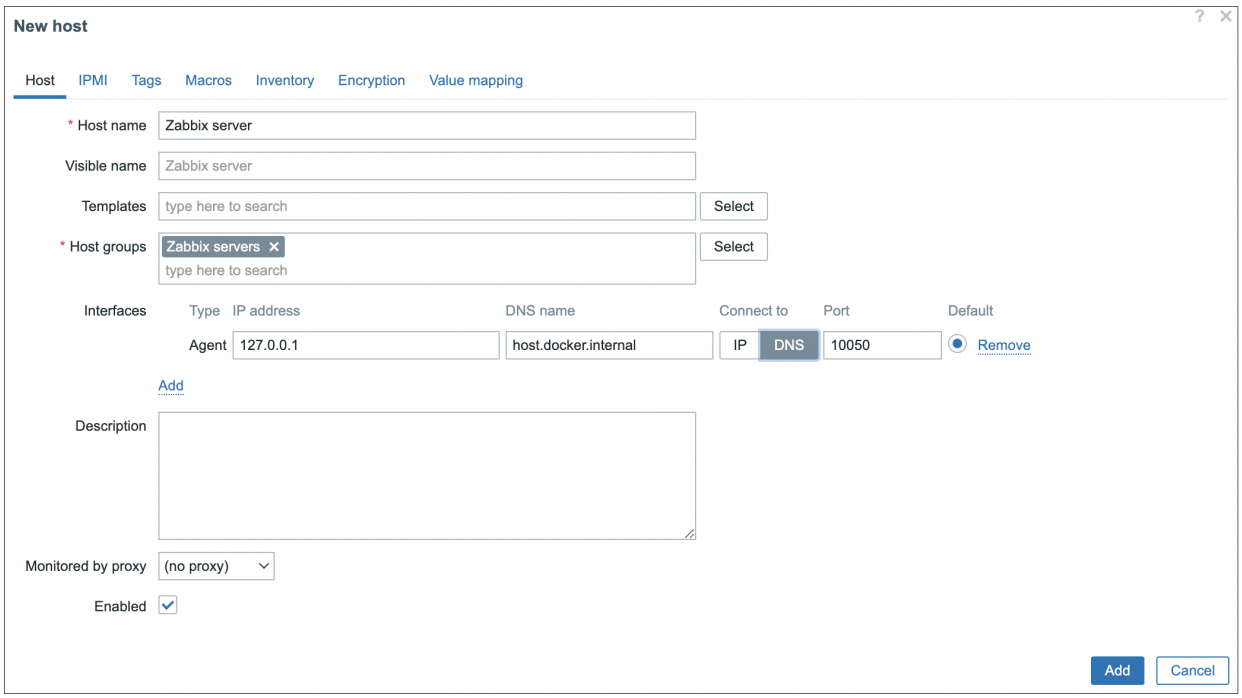
NOTE	По умолчанию: Логин: Admin Пароль: zabbix

2. Укажите место установки Zabbix Agent. Для этого выполните следующее:

   В боковом меню выберите **Monitoring** и затем нажмите кнопку **Create host** в правом верхнем углу страницы.



3. В открывшемся диалоговом окне задайте значения следующих параметров:
    - **Host name** – имя хоста (например, Zabbix server);
    - **Host groups** – группы хостов (нажмите кнопку **Select** и в открывшемся меню выберите **Zabbix-servers**);
    - В секции **Interfaces** нажмите кнопку **Add** и в открывшемся меню выберите пункт **Agent**;
    - В полях **IP address**, **DNS name**, **Connect to** и **Port** задайте параметры подключения агента.



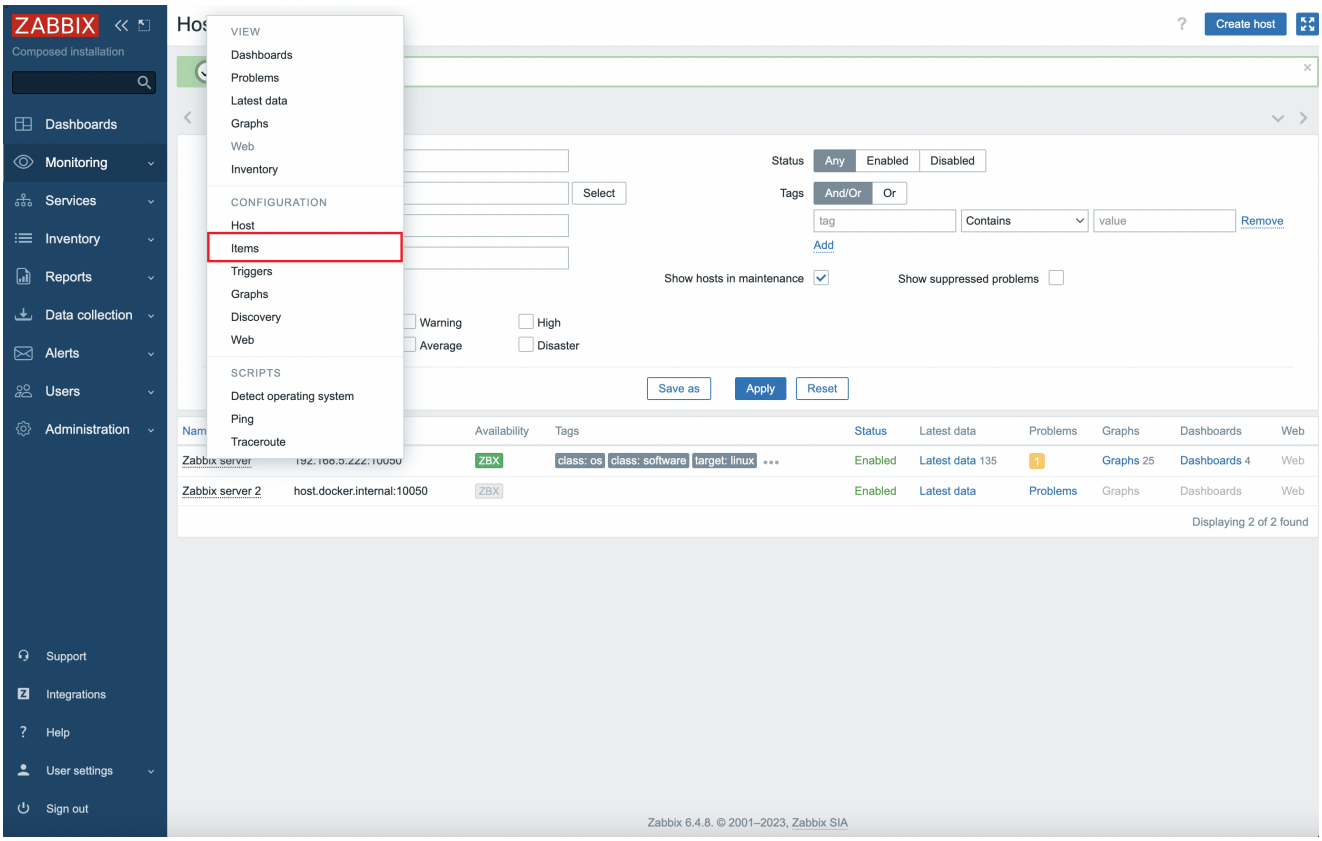
4. Нажмите кнопку **Add**.

В результате сервер Zabbix будет получать данные с агента.

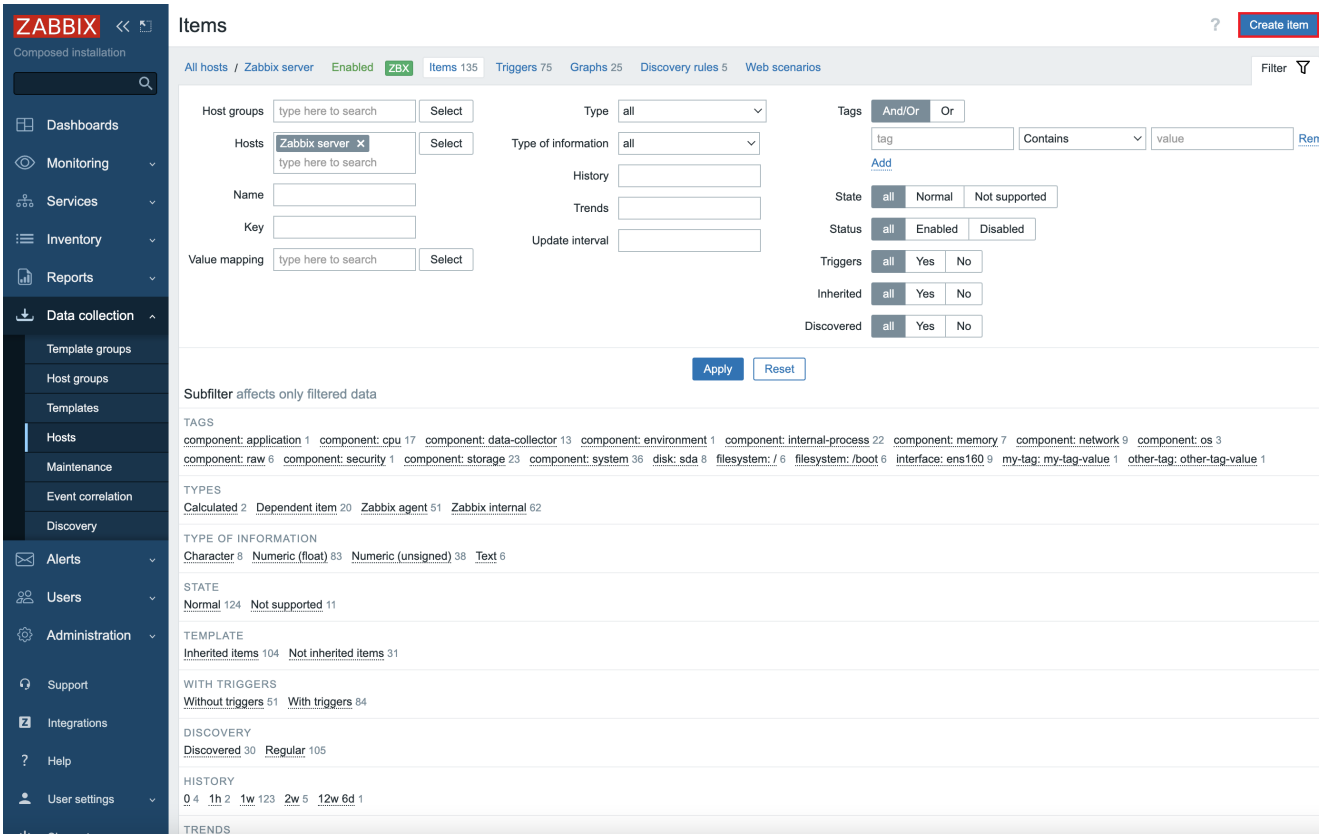
Шаг 5: создание объекта для мониторинга

Создайте новый объект (item) для мониторинга. Для этого выполните следующее:

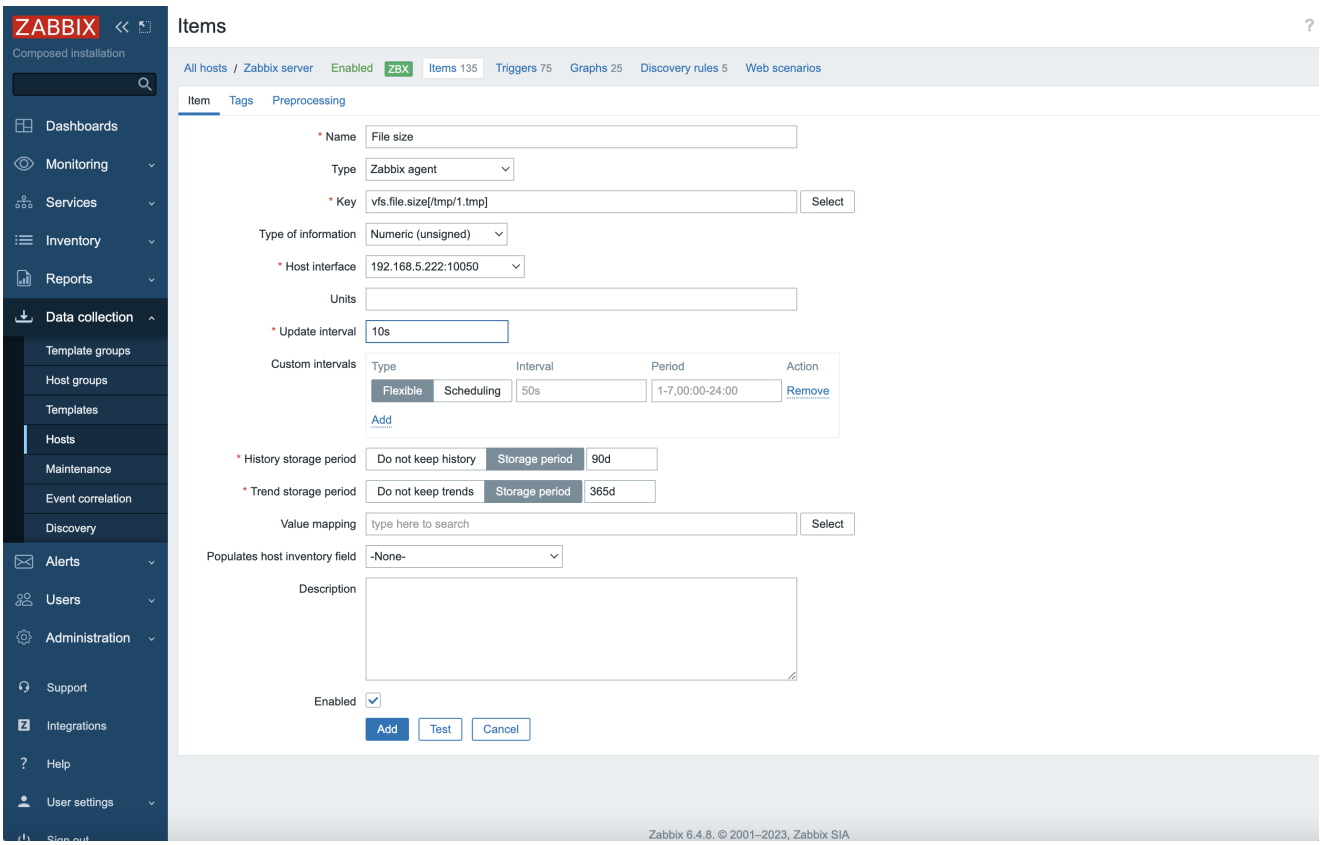
1. Нажмите на имя созданного сервера и в открывшемся меню выберите **Items**.



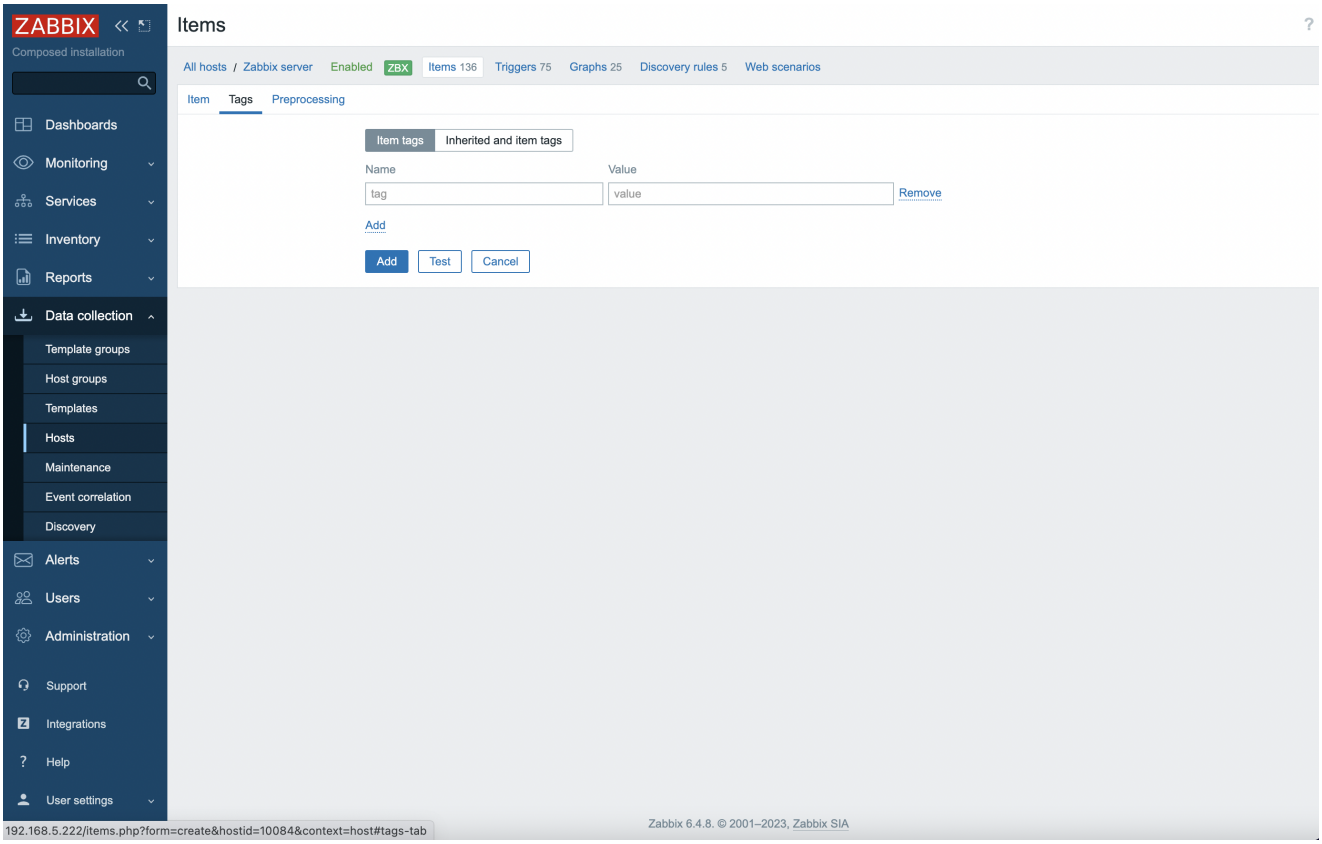
2. Нажмите кнопку **Create item** в правом верхнем углу станицы.



3. В открывшемся диалоговом окне задайте нужные параметры для item.



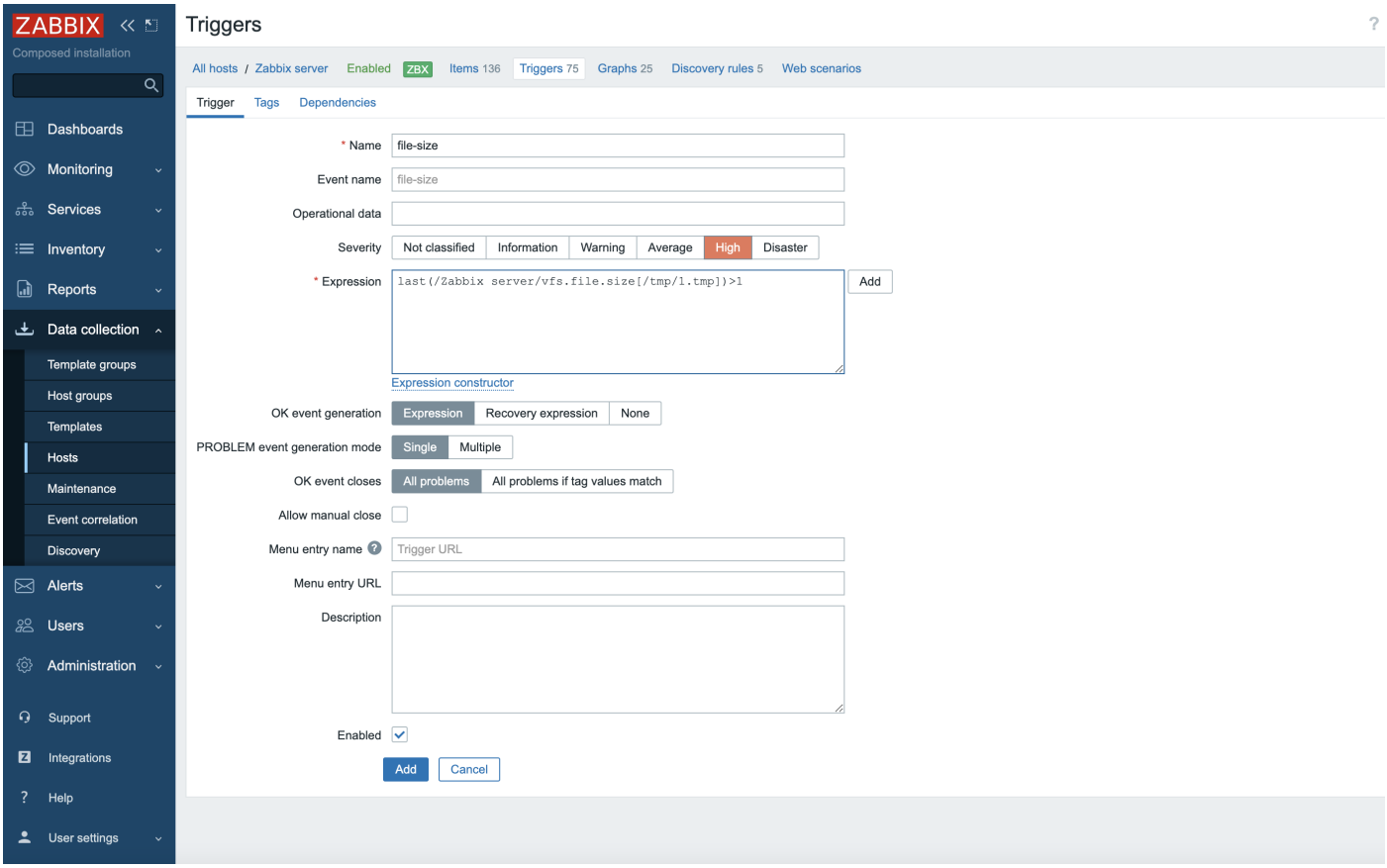
4. Переключитесь на вкладку **Tags**, задайте нужные теги и нажмите кнопку **Add**.



Шаг 6: создание триггера

Переключитесь на вкладку **Triggers** и нажмите кнопку **Create trigger**, чтобы создать новый триггер.

В качестве примера создайте триггер (например, **file-size-trigger**) на размер файла более 1 байта. После ввода параметров нажмите кнопку **Add**. Если размер файла превысит 1 байт, то возникнет авария уровня HIGH.

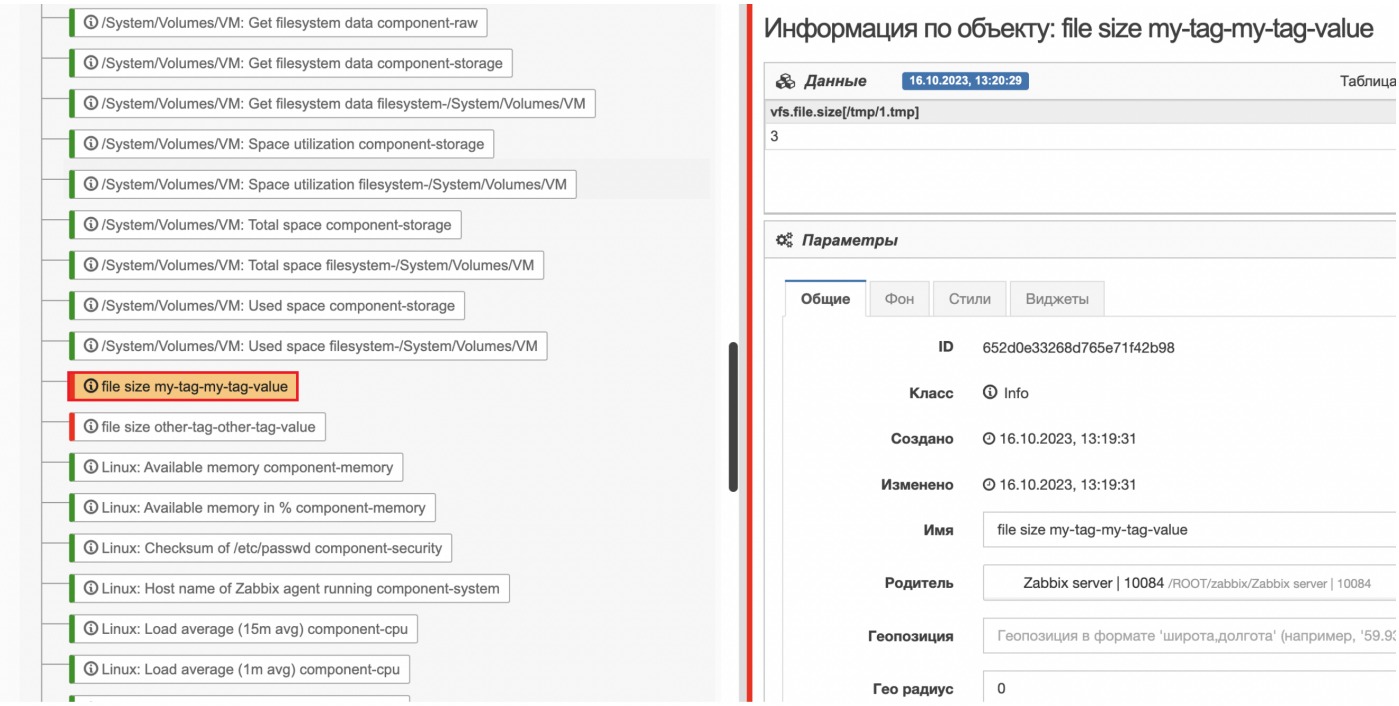


Шаг 7: настройка подключения к серверу Центрального пульта

Для подключения Центрального пульта к серверу Zabbix в конфигурационном файле **saymon-server.conf** укажите адрес сервера Zabbix и затем перезапустите сервер Центрального пульта (подробнее см. Конфигурация сервера).

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

В списке объектов Центрального пульта отобразятся Zabbix-хост и его дочерние объекты.



Шаг 8: управление событием

Для создания аварии на хосте выполните команду:

```
$ echo -n "create an accident" > /home/zabbix/test-file.txt
```

Для устранения аварии на хосте используйте команду:

```
$ echo -n "clean" > /home/zabbix/test-file.txt
```

3.5.1.3. Периодичность обновления данных

По умолчанию данные на сервере системы обновляются каждые 2 минуты.

Для изменения периода обновления данных необходимо в раздел **Zabbix** конфигурационного файл сервера `/etc/saymon/saymon-server.conf` добавить параметр `polling_period` со значением периода обновления данных в миллисекундах. Например:

```
"zabbix" : [{
    "url" : "http://192.168.1.215/zabbix/api_jsonrpc.php",
    "user" : "saymon",
    "password" : "saymon_user_password",
    "parent_id" : "58b586d5c3a2f96642e25537",
    "polling_period": 30000
}]
```

Для применения настроек необходимо перезапустить сервер Центрального Пульта:

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

3.5.1.4. Изменение стандартных опций в запросах Zabbix

Возможно изменять опции для запросов

- хостов (hosts),
- триггеров (triggers),
- данных (data).

NOTE

Все возможные поля фильтров представлены в документации по Zabbix API:

- hosts,
- triggers,
- data.

```
"zabbix" : [{
...
       "request_options": {
          "hosts": {
            "filter":  {
               ...
            }
          },
          "triggers": {
            "filter": {
                ...
             }
           },
           "data": {
            "filter": {
                ...
             }
           }
        }
...
}]
```

NOTE

Чтобы убрать фильтр из опции запроса, необходимо указать параметру **filter** значение **null**.

Пример (убрать поле **filter** из опций запроса триггеров Zabbix):

```
"zabbix" : [{
...
```

```
        "triggers": {
            "filter": null
        }
...
}]
```

3.5.1.5. Корреляция класса создаваемых объектов с их именами

Обнаруженным объектам по умолчанию присваивается класс `INFO`. Чтобы присвоить новым объектам другие классы, необходимо задать массив **zabbix.classes** в конфигурационном файле сервера `/etc/saymon/saymon-server.conf` и указать в нём соответствия идентификаторов классов и имён объектов:

```
"zabbix" : [{
...
        "classes" : {
            "4" : "CPU",
            "619503be0ffb595aebe22222" : "Memory|General"
         },
...
}]
```

CAUTION	Порядок применения правил привязки не гарантируется. Если назначить двум разным классам одинаковые регулярные выражения или использовать выражение **"."*** (любой текст), то нет гарантии, что сначала будет применено правило, указанное выше в списке.

Для каждого Zabbix-подключения массив настраивается индивидуально. Регулярное выражение по имени объекта нечувствительно к регистру символов.

CAUTION	При необходимости связать класс с несколькими именами нужно указать все имена в одном регулярном выражении с использованием оператора ИЛИ. В противном случае применится только одно соответствие. Пример: **"619503be0ffb595aebe22222": "Memory|General"**

3.5.1.6. Логирование

Чтобы выводить параметры и результаты запросов в лог, необходимо задать параметр **zabbix.debug** со значением **true**:

```
"zabbix" : [{
```

```
    ...
    "debug": true,
    ...
    }
]
```

3.5.2. Интеграция с Grafana

Grafana – популярный инструмент визуализации данных, позволяющий отображать графики и диаграммы на удобных визуальных дашбордах.

Центральный Пульт интегрируется с Grafana через специальный datasource-плагин, который позволяет отображать на диаграммах данные по указанным метрикам из указанных объектов Центрального Пульта. При помощи стандартного инструментария Grafana данные визуализируются, и на их основании строятся дашборды.

Ниже приведена последовательность шагов для добавления данных по метрике из Центрального Пульта в Grafana. Все действия, связанные с интеграцией, осуществляются в Grafana.

1. Авторизоваться в Grafana.
2. С главной страницы рабочего пространства перейти в **Configurations - Data Sources** через боковое меню и добавить data source (источник данных) нажатием на соответствующую кнопку.

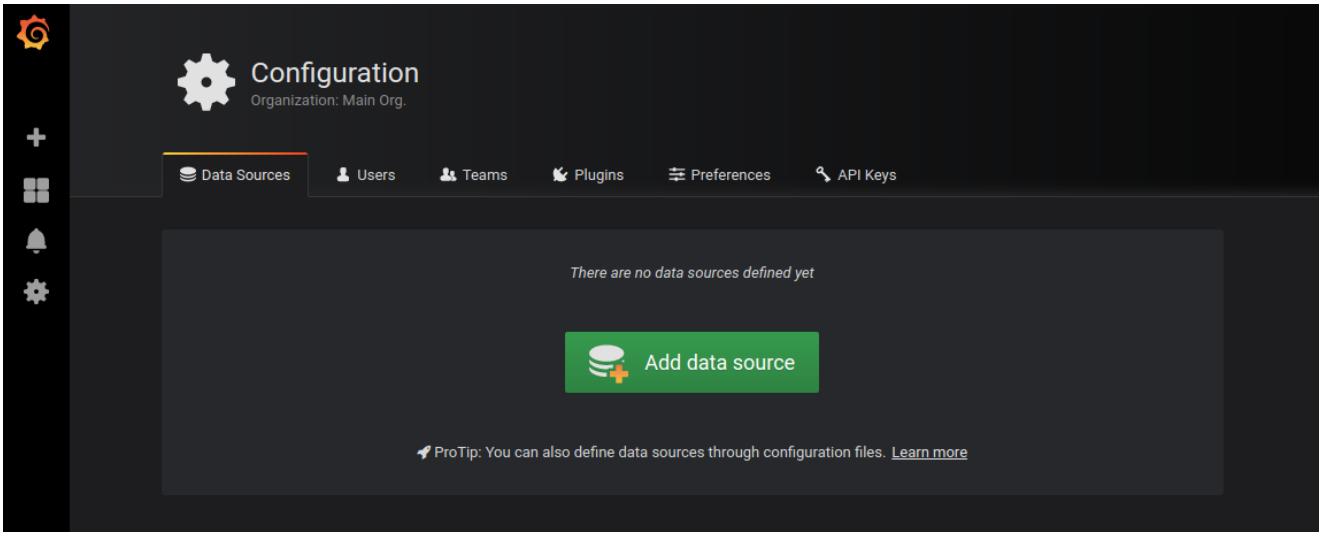


Рис. 7. Добавление источника данных в Grafana

3. В открывшемся окне типов источников данных выбрать SAYMON.

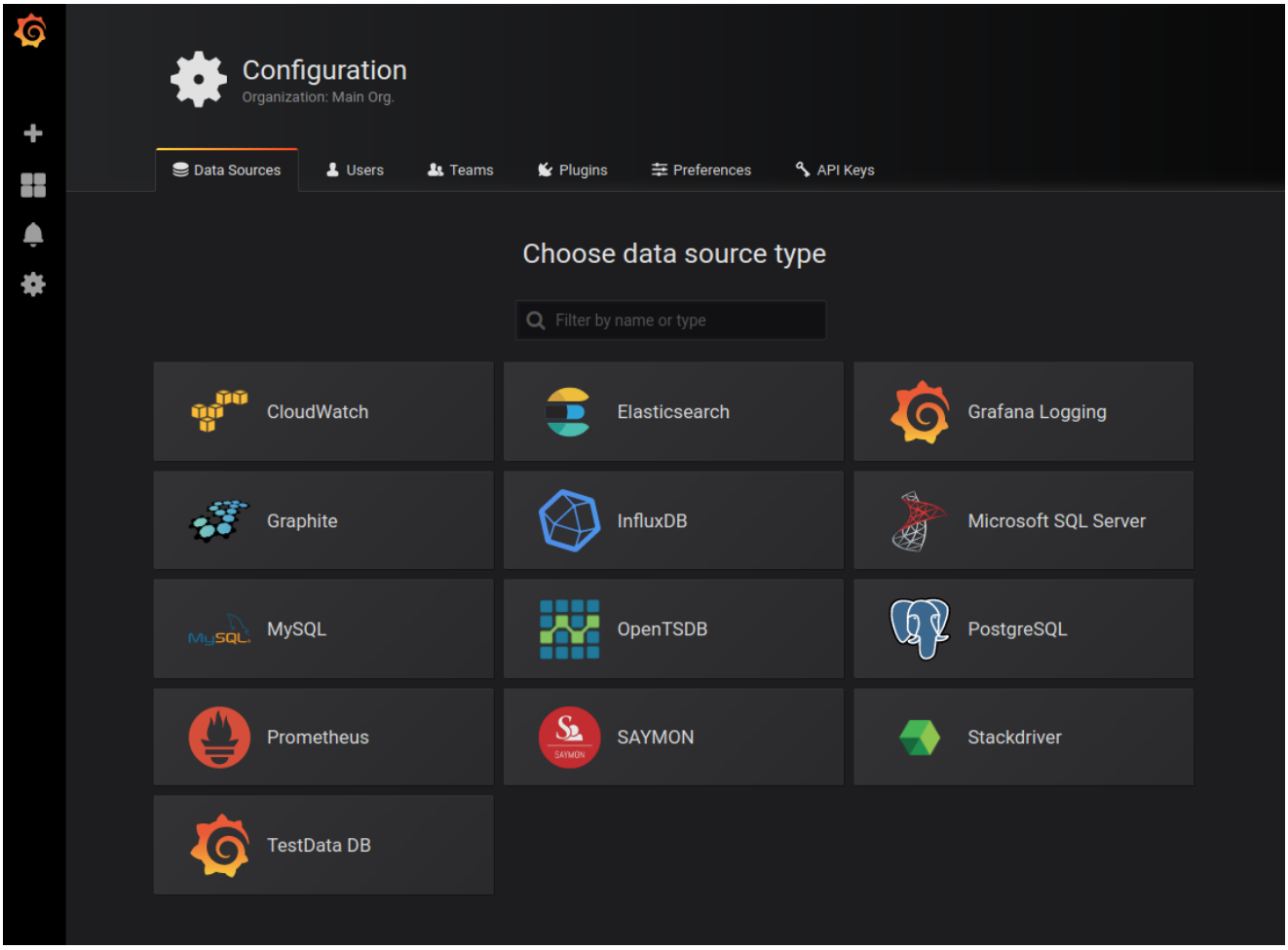


Рис. 8. Выбор источника данных в Grafana

4. В окне настроек необходимо указать:
    - имя источника данных;
    - URL-адрес, откуда будут собираться данные;
    - режим доступа, как будут обрабатываться запросы к источнику данных: **server** или **browser**;
    - имена cookies-файлов, которые должны быть перенаправлены в источник данных;
    - тип аутентификации;
    - логин и пароль, используемые в Центральном Пульте.

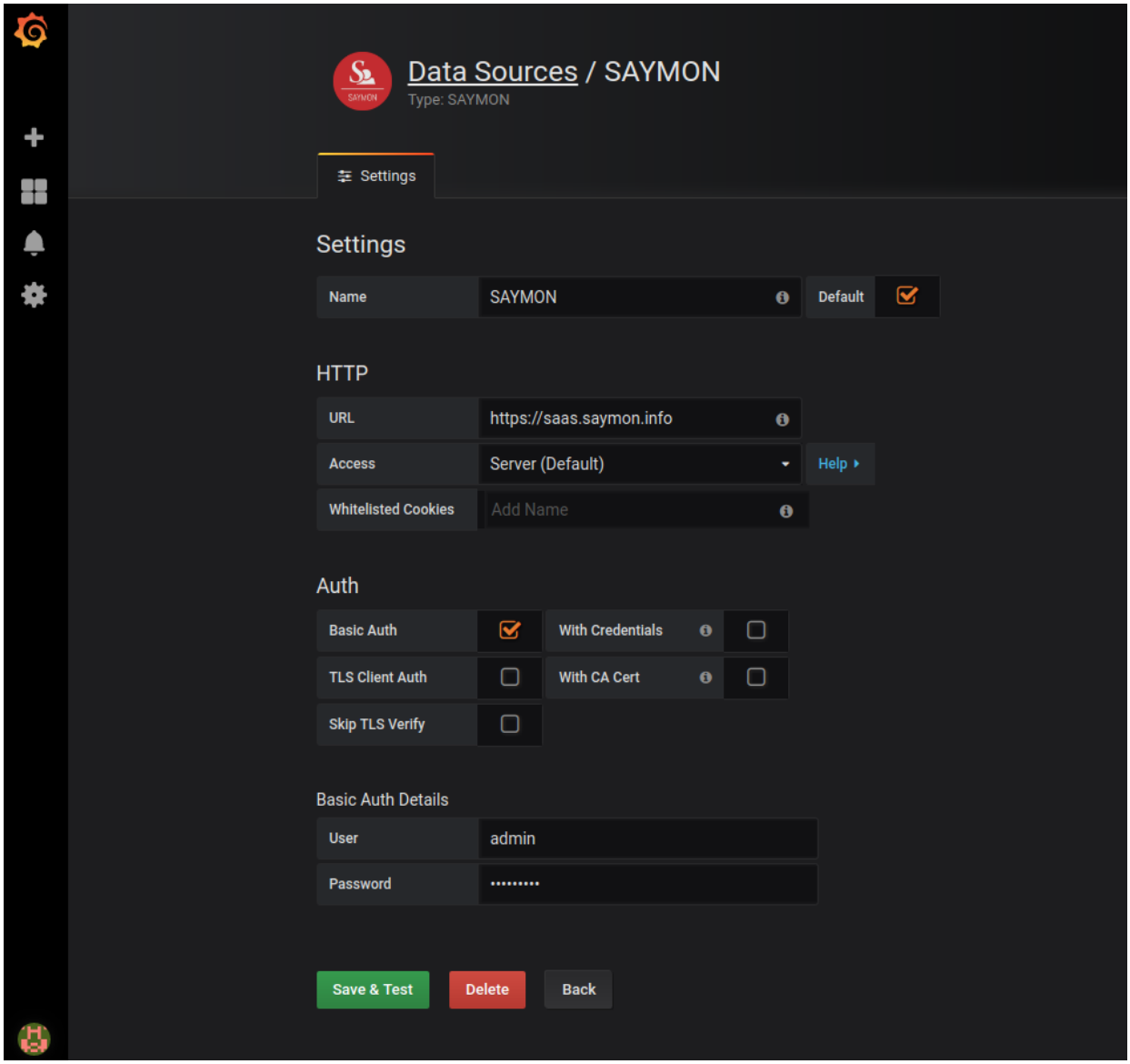


Рис. 9. Настройка источника данных в Grafana

5. Через боковое меню перейти в окно управления дашбордами (**dashboards - manage**) и приступить к созданию нового дашборда нажатием соответствующей кнопки.

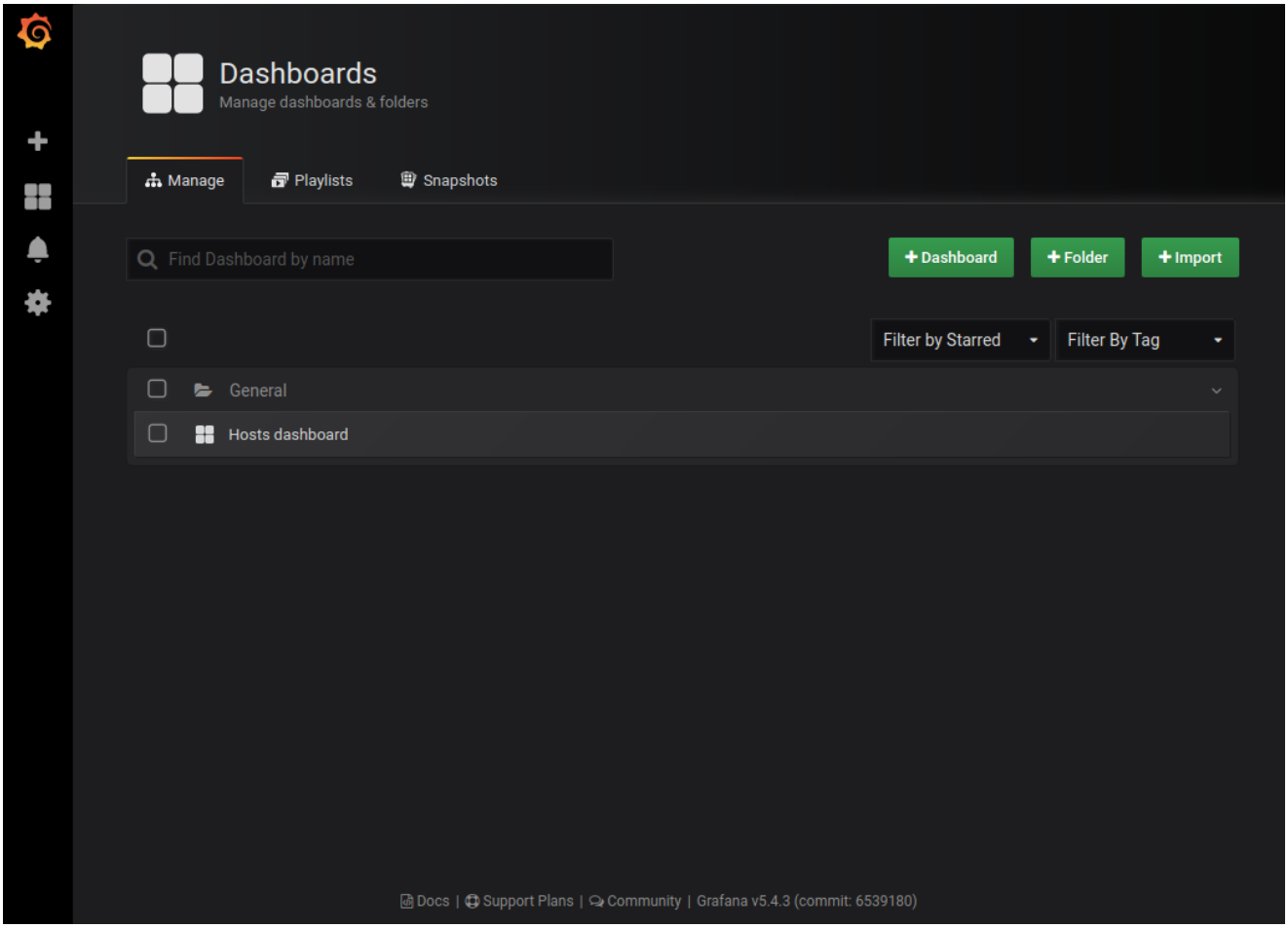


Рис. 10. Создание нового дашборда в Grafana

6. Выбрать тип отображения дашборда. Например, график.



Рис. 11. Выбор типа отображения дашборда в Grafana

7. Открыть контекстное меню нажатием на стрелку возле заголовка панели и перейти в режим редактирования.
8. В окне редактирования перейти во вкладку **Metrics** и добавить объекты, к которым у пользователя есть доступ в Центральном Пульте. Указать:
   - имя или ID объекта;
   - метрику, которую необходимо отобразить на графике;
   - отображать ли путь до объекта.

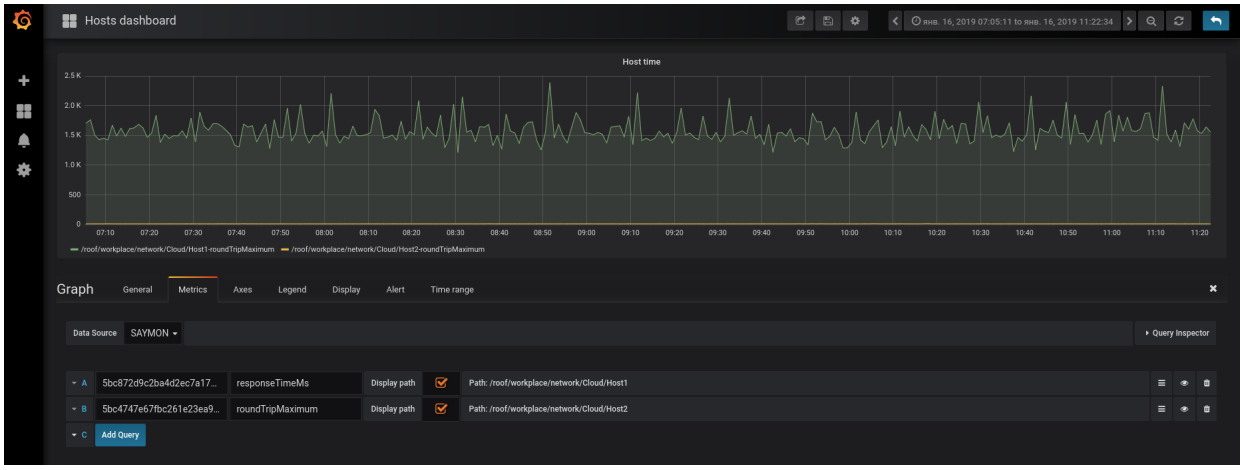


Рис. 12. Вкладка Metrics в Grafana

9. Настроить временной период, за который данные отображаются на графике, и интервал их обновления.

3.5.3. Интеграция с Keycloak

Для пользователей Центрального Пульта предусмотрена возможность авторизации посредством сервера Keycloak.

3.5.3.1. Настройка сервера

Для настройки взаимодействия между сервером Центрального Пульта и внешним сервером Keycloak необходимо получить хотя бы один из параметров:

- Realm Certificate, которым сервер Keycloak подписывает пользовательские токены,
- файл настроек соединения для Backend-клиента.

WARNING	Настройка клиента и получение параметров осуществляются в консоли администратора Keycloak.

Realm Certificate

Для получения Realm Certificate необходимо:

1. Выбрать требуемую область (Realm) из списка (здесь - **SAYMON**):

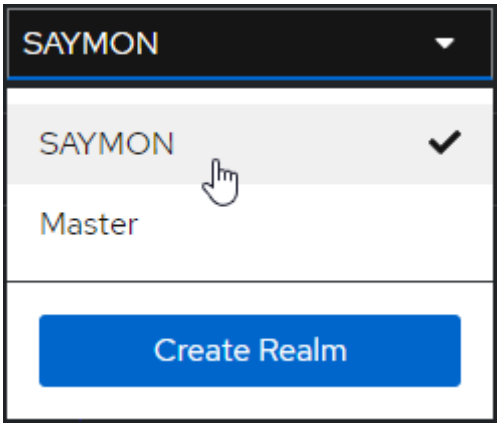


Рис. 13. Выбор области в Keycloak

2. В секции **Realm settings** на вкладке **Tokens** уточнить алгоритм подписи токенов в поле **Default Signature Algorithm** (здесь - **RS256**):

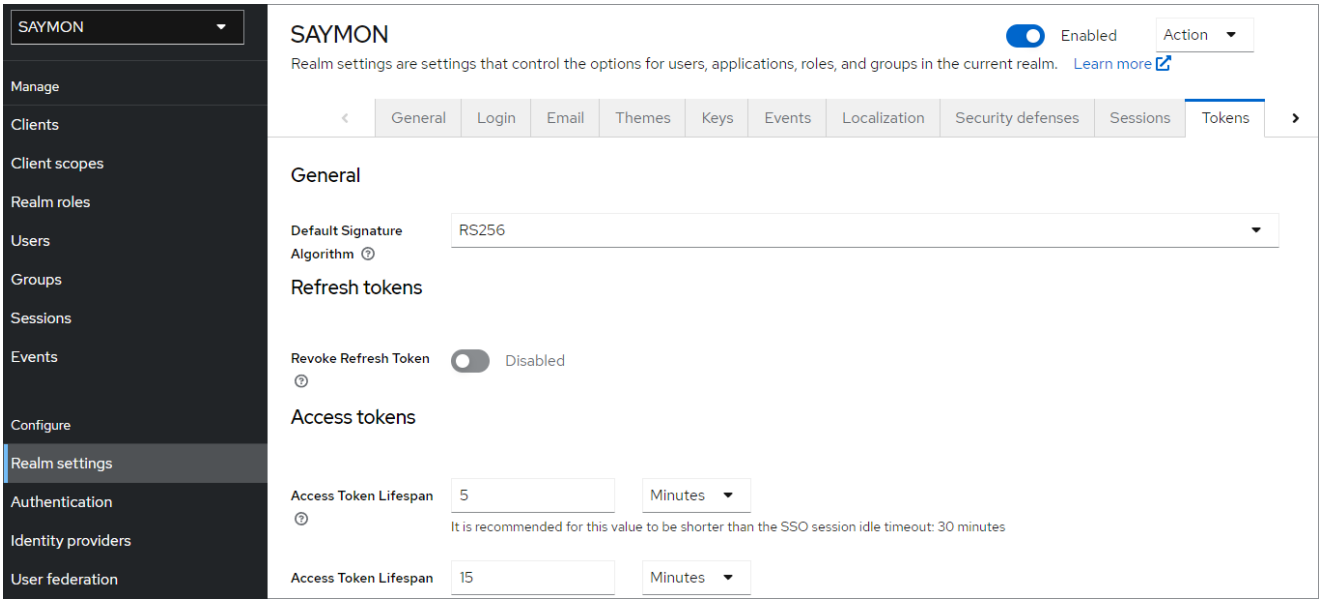


Рис. 14. Выбор алгоритма подписи токенов

3. На вкладке **Keys** найти сертификат для данного алгоритма:

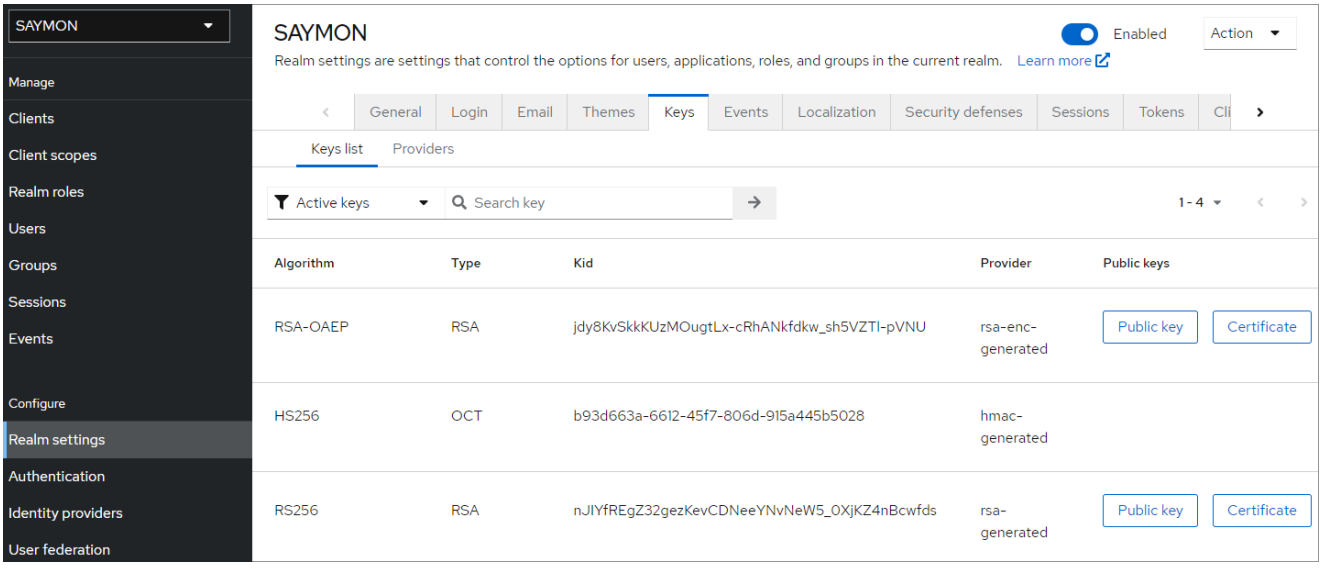


Рис. 15. Вкладка Keys

4. Кнопка Certificate в соответствующей строке открывает искомый сертификат:

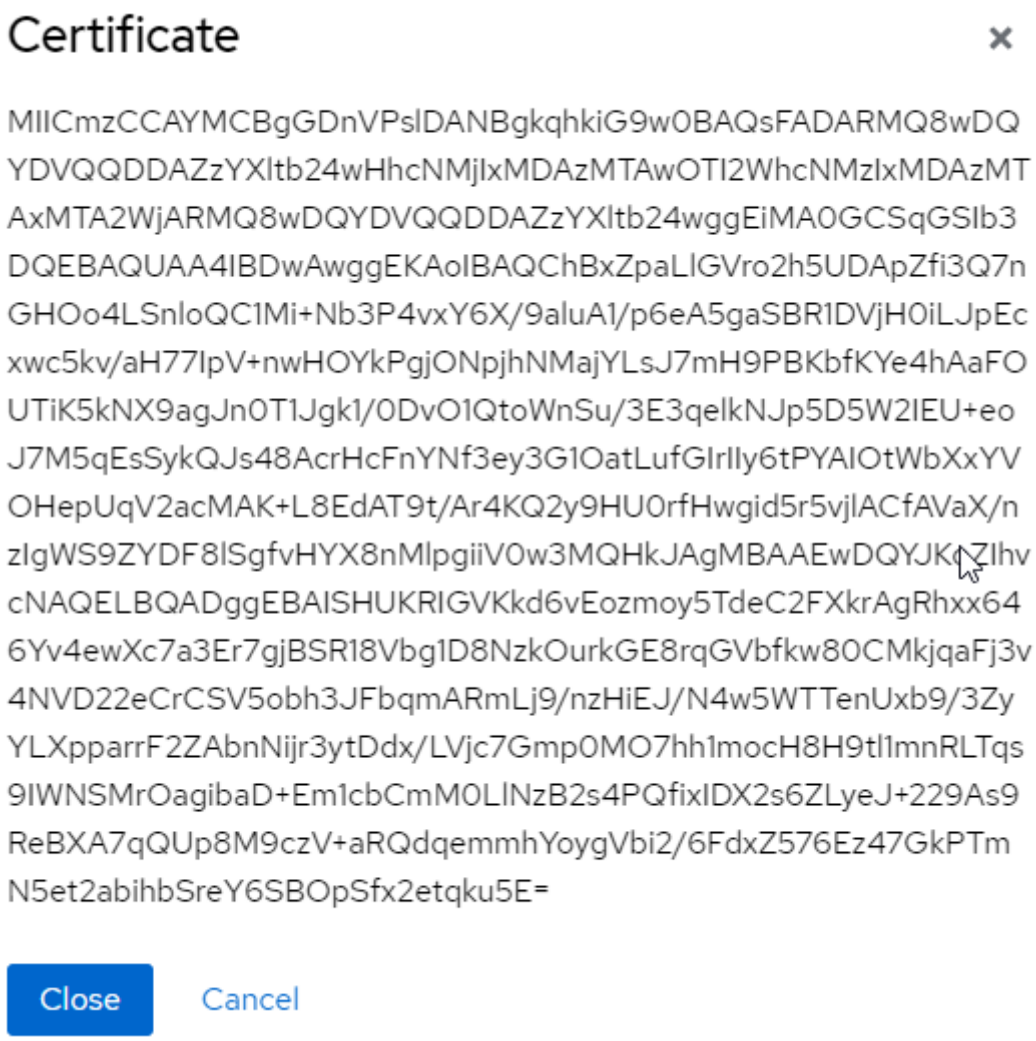


Рис. 16. Сертификат ключа на вкладке Keys

Данные рекомендуется сохранить в файл **keycloak_realm_certificate**.

Backend-клиент

Клиенты - это параметры соединения с Keycloak-сервером для внешних приложений, которые авторизуют пользователей.

Главной особенностью Backend-клиента является наличие собственной аутентификации: такой клиент получает конфиденциальный доступ по секретному ключу.

Для настройки Backend-клиента необходимо:

1. Выбрать требуемую область (Realm) из списка (здесь - **SAYMON**):

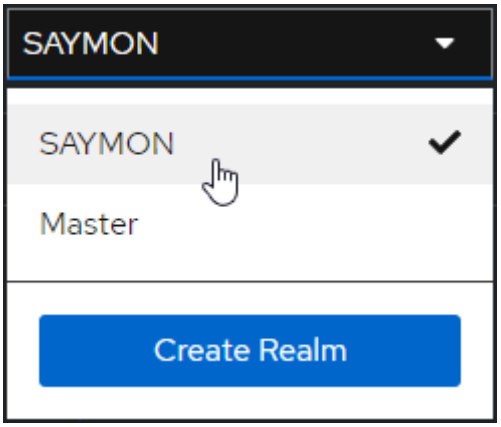


Рис. 17. Выбор области в Keycloak

2. В секции **Clients** нажать кнопку Create client.

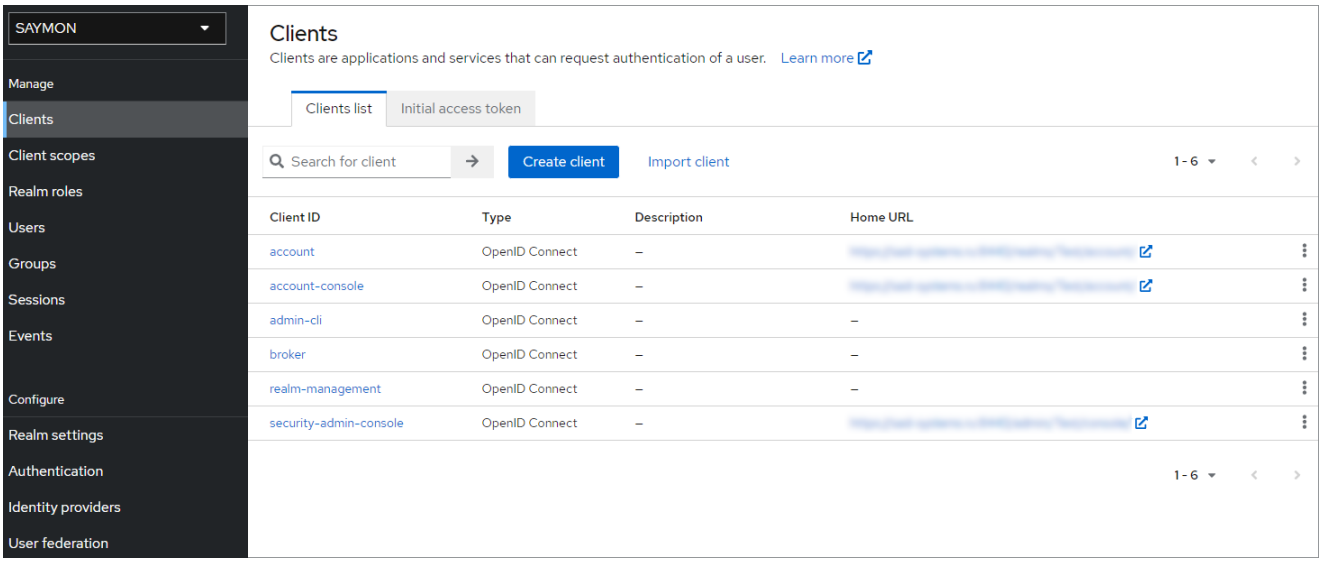


Рис. 18. Секция Clients

3. Ввести имя Backend-клиента в поле **Client ID**:

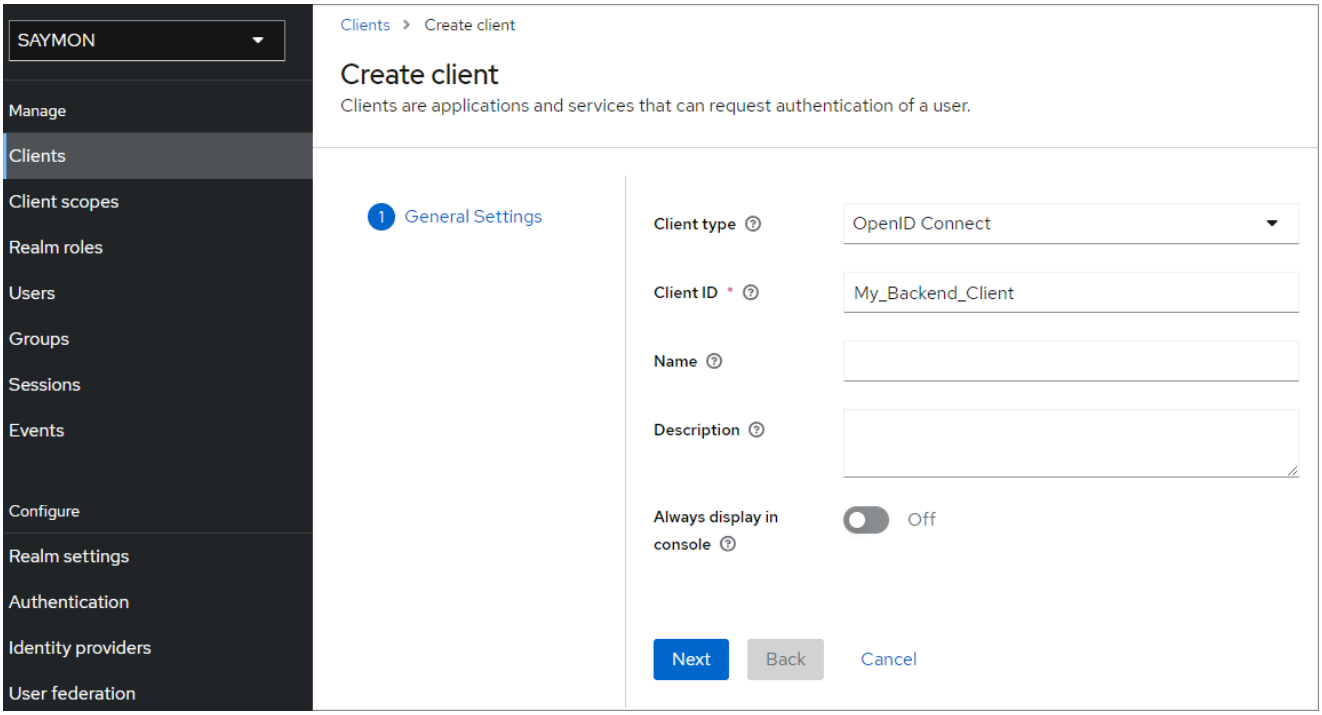


Рис. 19. Создание клиента, вкладка General Settings

и нажать кнопку Next.

4. Слайдер **Client authentication** установить в положение **On**:

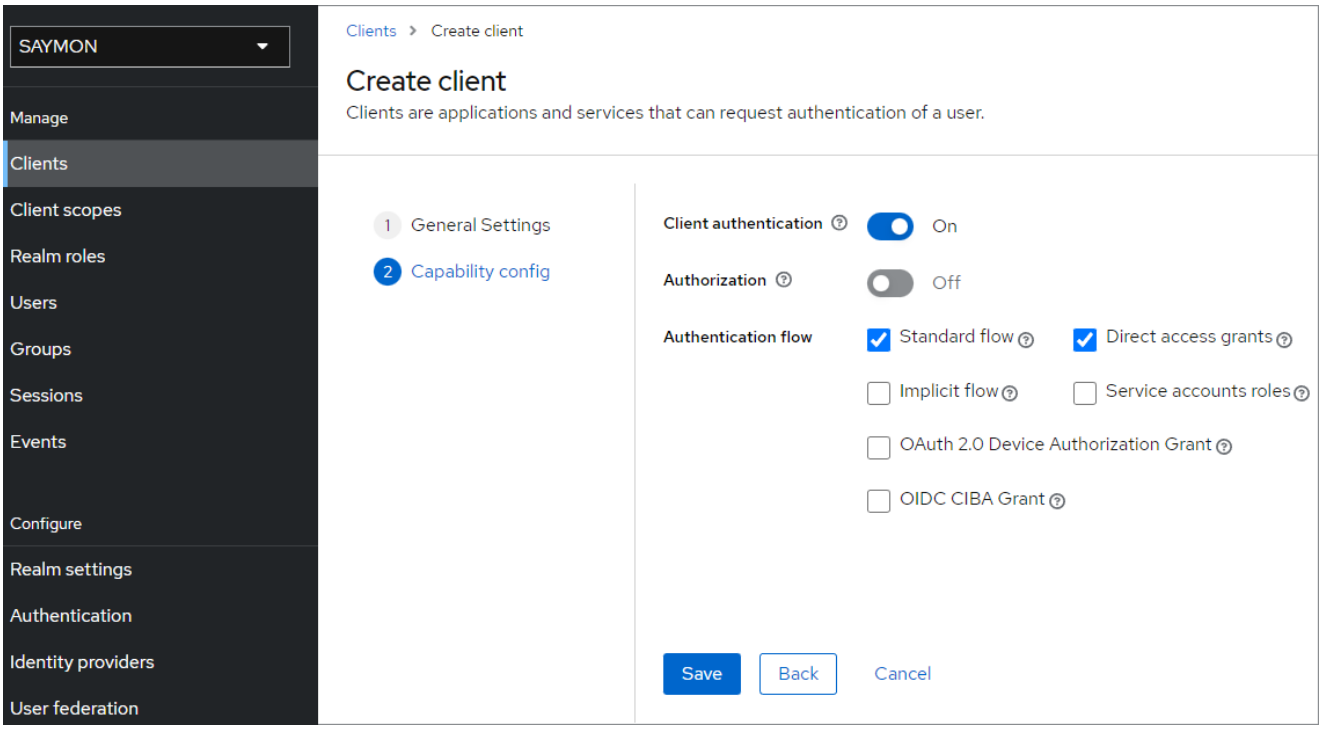


Рис. 20. Создание клиента, вкладка Capability config

и нажать кнопку Save.

5. В списке **Action** выбрать **Download adapter config**:

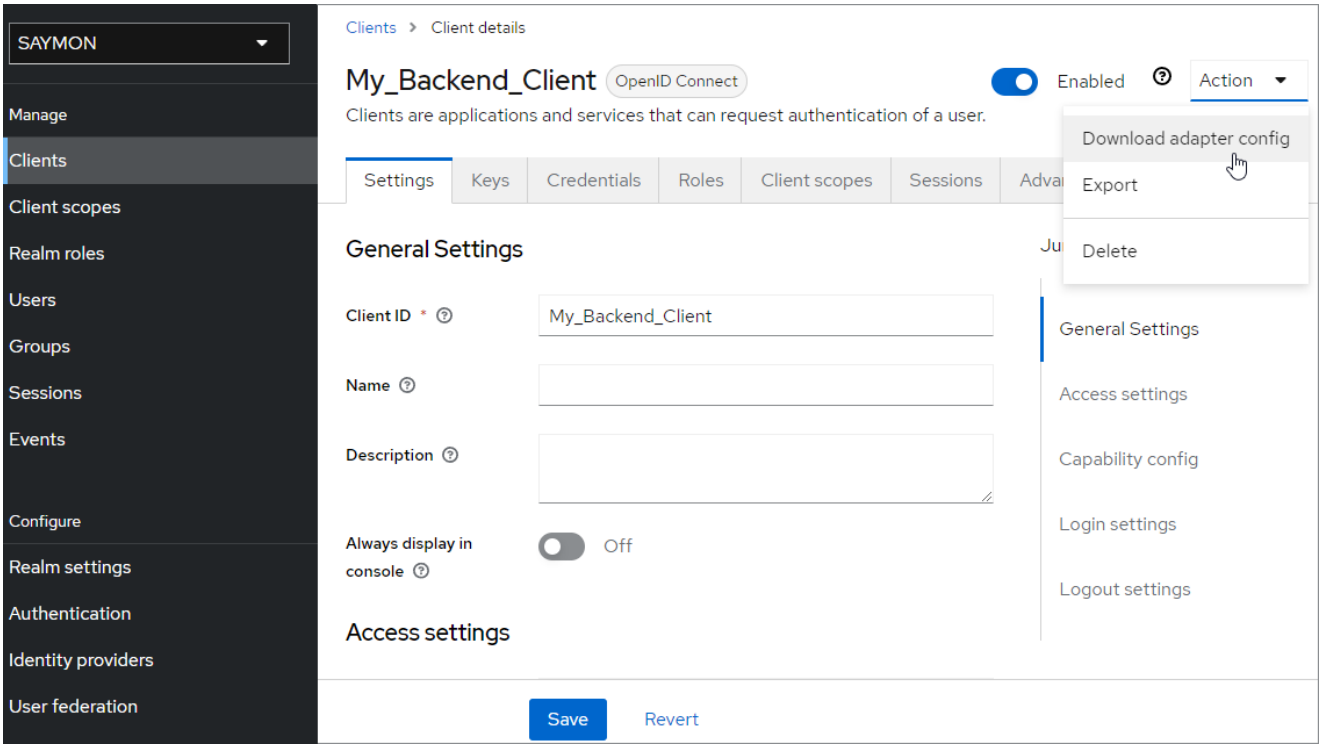


Рис. 21. Кнопка Download adapter config

6. Нажать кнопку Download, чтобы получить файл настроек соединения для Backend-клиента в формате JSON:

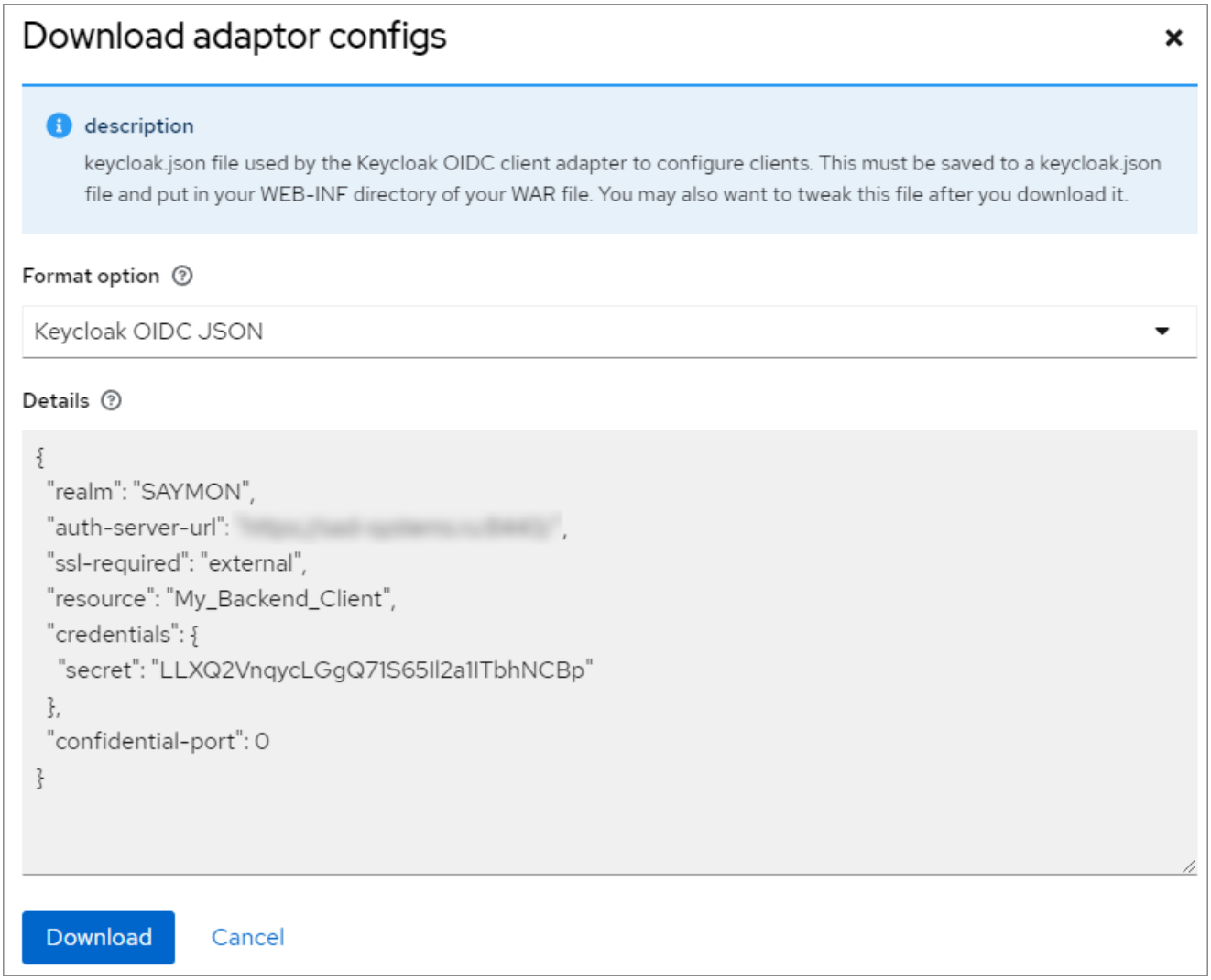


Рис. 22. Файл настроек соединения для Backend-клиента

Конфигурация сервера Центрального Пульта

Для взаимодействия с сервером Keycloak на сервере Центрального Пульта необходимо:

1. Добавить раздел **Keycloak** в конфигурационный файл сервера `/etc/saymon/saymon-server.conf`:

```
"keycloak": {
  "realm_certificate": "<данные keycloak_realm_certificate>",
  "realm_certificate_file": "/etc/saymon/keycloak_realm_certificate",
    "config_file": "/etc/saymon/keycloak_backend.json"
}
```

   где:

   - **realm_certificate** - данные Keycloak Realm Certificate;
   - **realm_certificate_file** - путь к файлу с Keycloak Realm Certificate;
   - **config_file** - путь к файлу настроек соединения, сгенерированному сервером Keycloak для Backend-клиента.

WARNING	Параметры перечислены в порядке понижения приоритета при одновременном указании. Для работы с Keycloak достаточно указать один из параметров.

2. Перезапустить службу saymon-server:

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

3.5.3.2. Настройка web-интерфейса

Для возможности авторизации пользователей посредством Keycloak через web-интерфейс Центрального Пульта необходимо получить файл настроек соединения Frontend-клиента.

WARNING	Настройка клиента и получение файла настроек соединения осуществляются в консоли администратора Keycloak.

Frontend-клиент

Главной особенностью Frontend-клиента является отсутствие собственной аутентификации: данный клиент получает публичный доступ.

Для настройки Frontend-клиента необходимо:

1. Выбрать требуемую область (Realm) из списка (здесь - **SAYMON**):

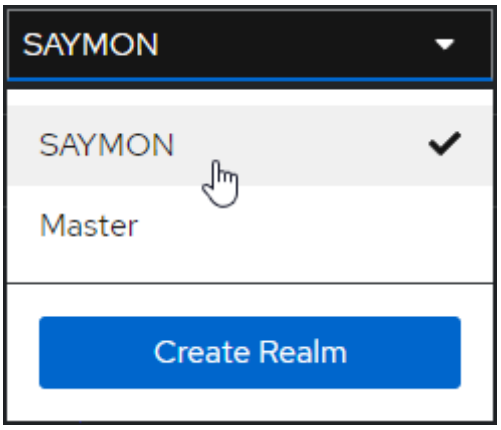


Рис. 23. Выбор области в Keycloak

2. В секции **Clients** нажать кнопку Create client:

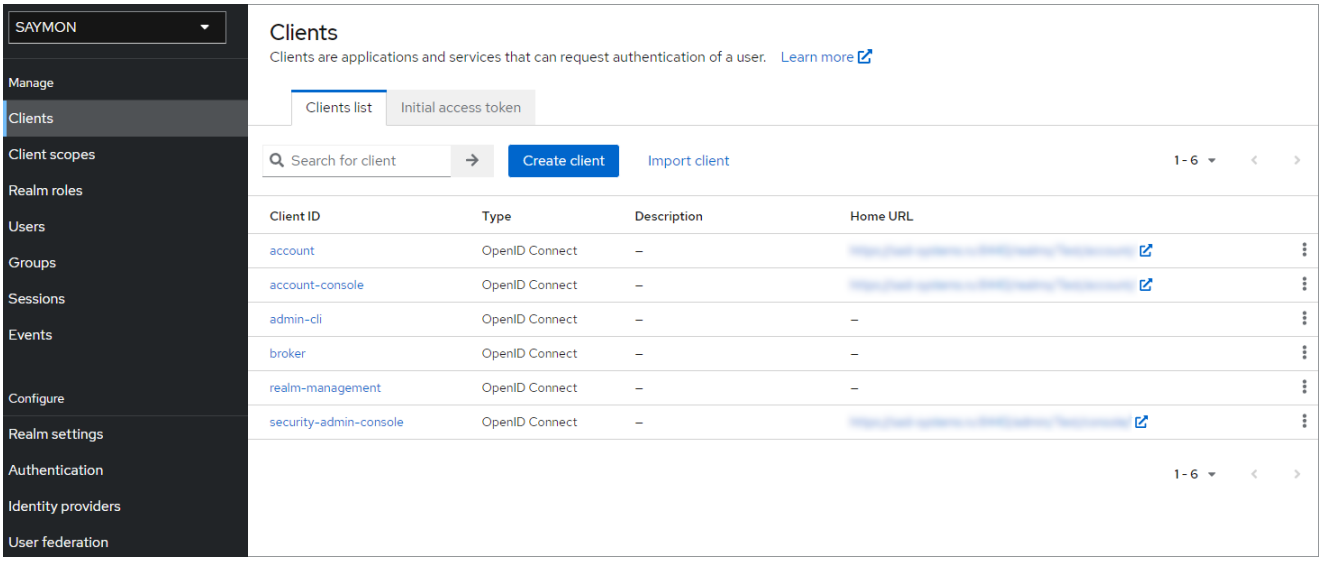


Рис. 24. Секция Clients

3. Ввести имя Frontend-клиента в поле **Client ID**:

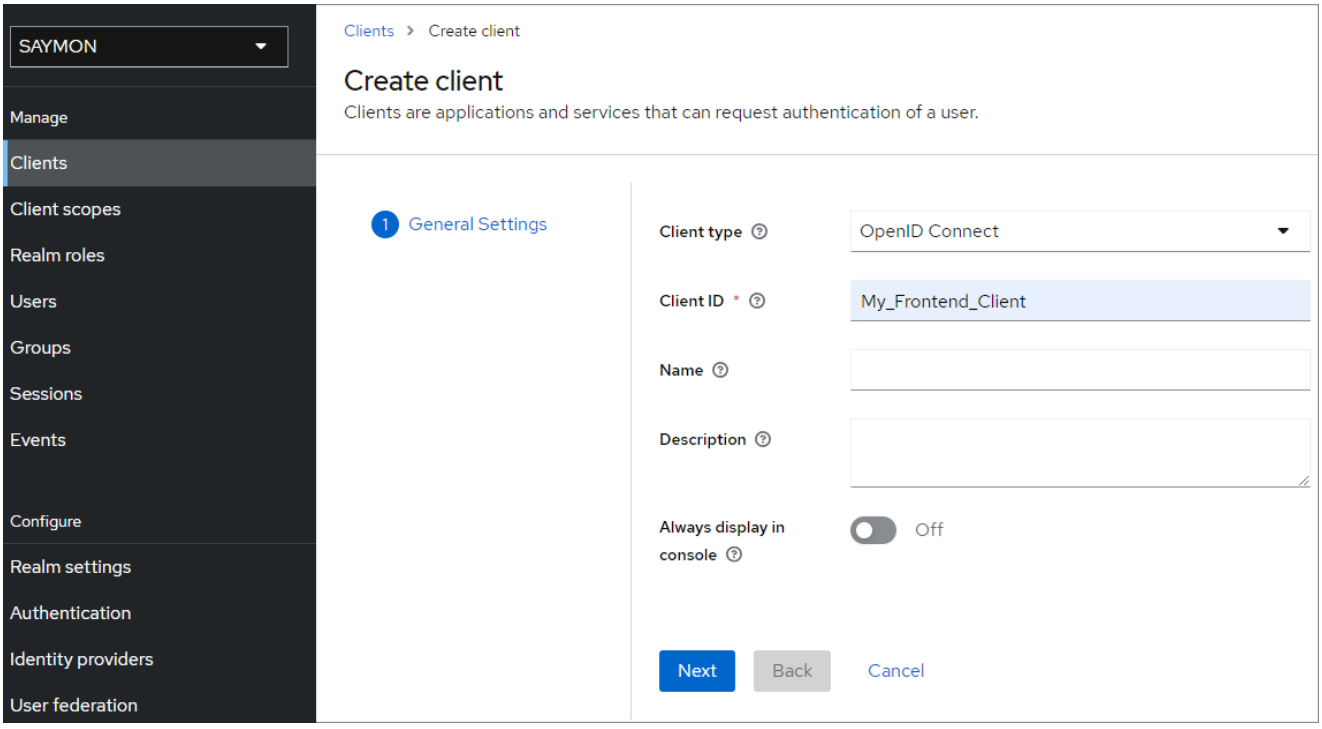


Рис. 25. Создание клиента, вкладка General Settings

и нажать кнопку Next.

4. Слайдер **Client authentication** установить в положение **Off**:

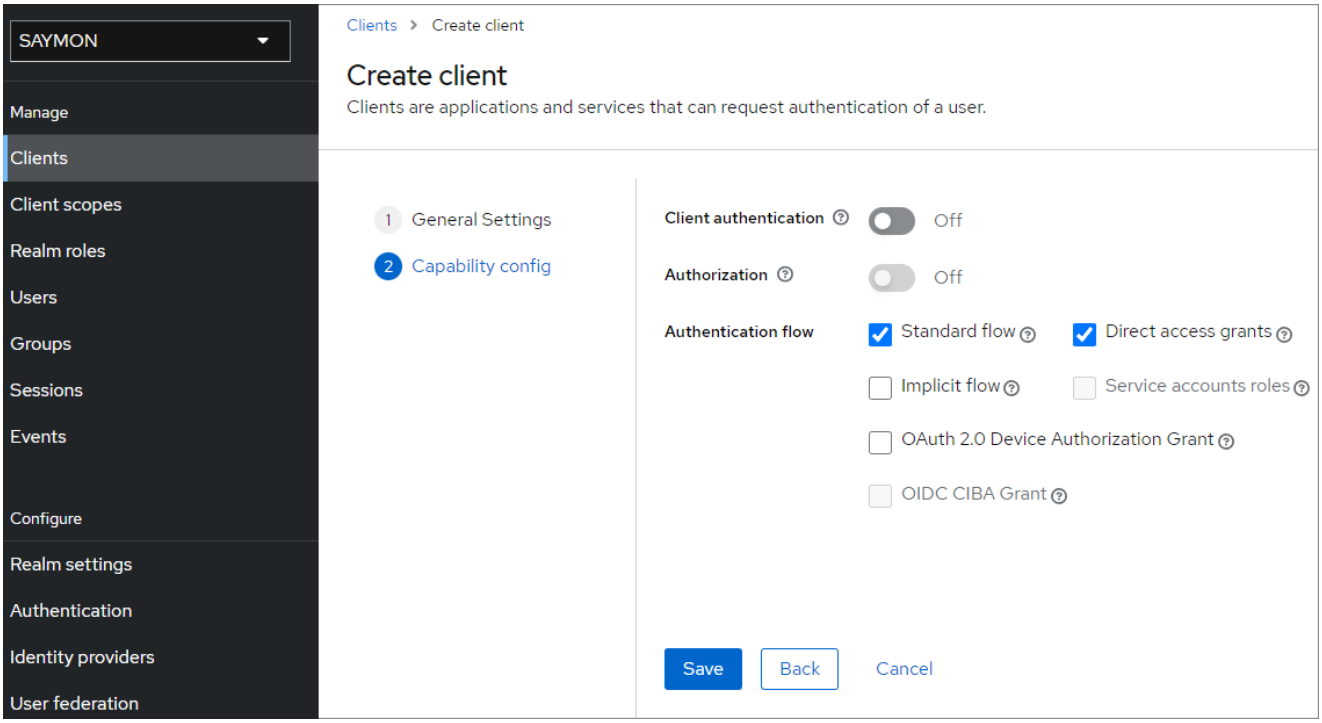


Рис. 26. Создание клиента, вкладка Capability config

и нажать кнопку Save.

5. На вкладке **Settings** в разделе **Access settings** указать корректные URI-адреса, с которых будет осуществляться доступ к Keycloak-серверу, и на которые будут перенаправляться пользователи после успешной аутентификации и выхода:

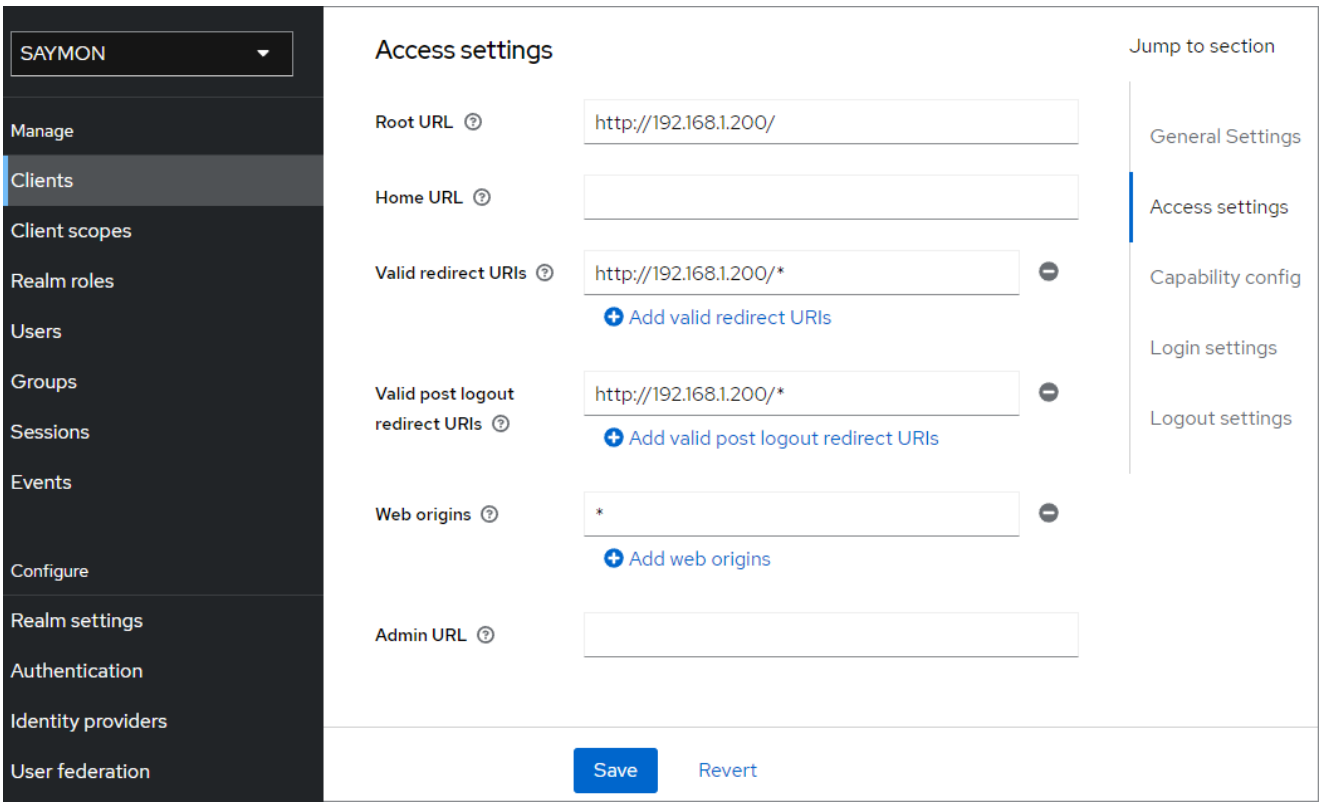


Рис. 27. Раздел Access Settings

и нажать кнопку Save.

6. В списке **Action** выбрать **Download adapter config**:

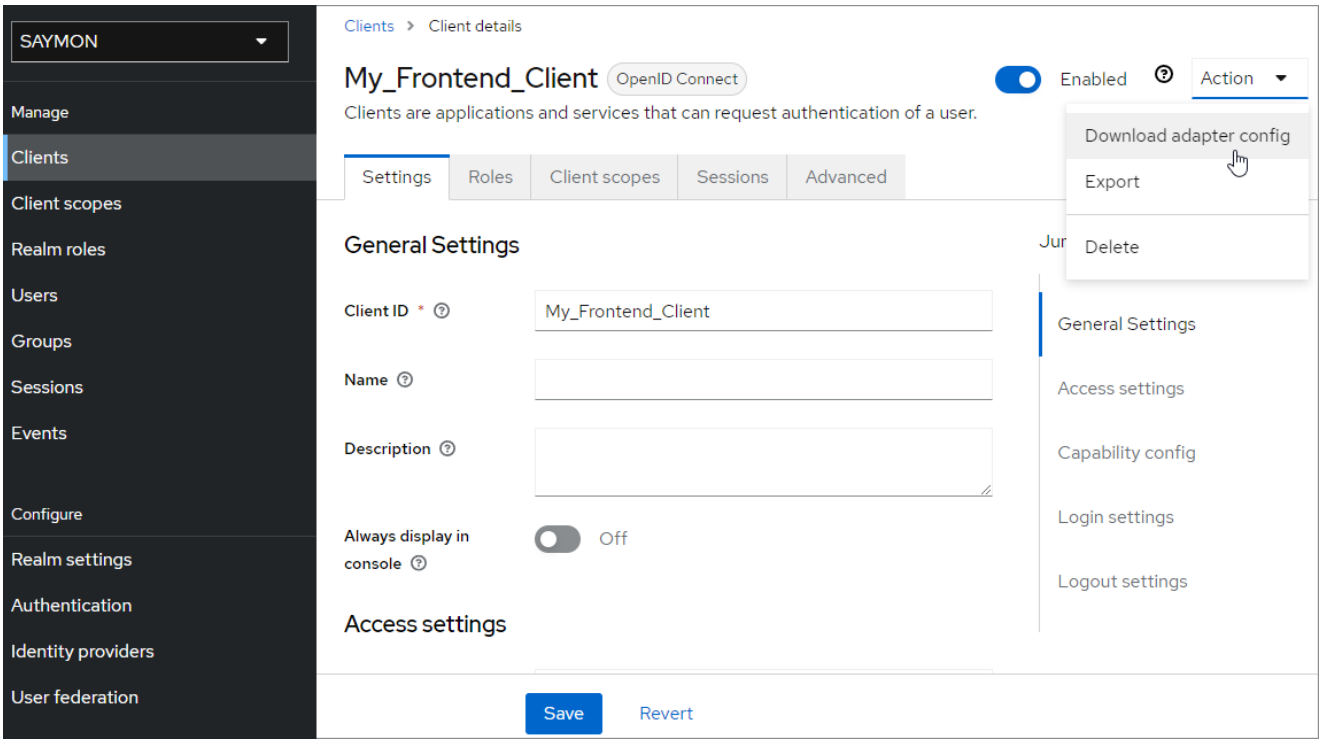


Рис. 28. Кнопка Download adapter config

7. Нажать кнопку Download, чтобы получить файл настроек соединения для Frontend-клиента в формате JSON:

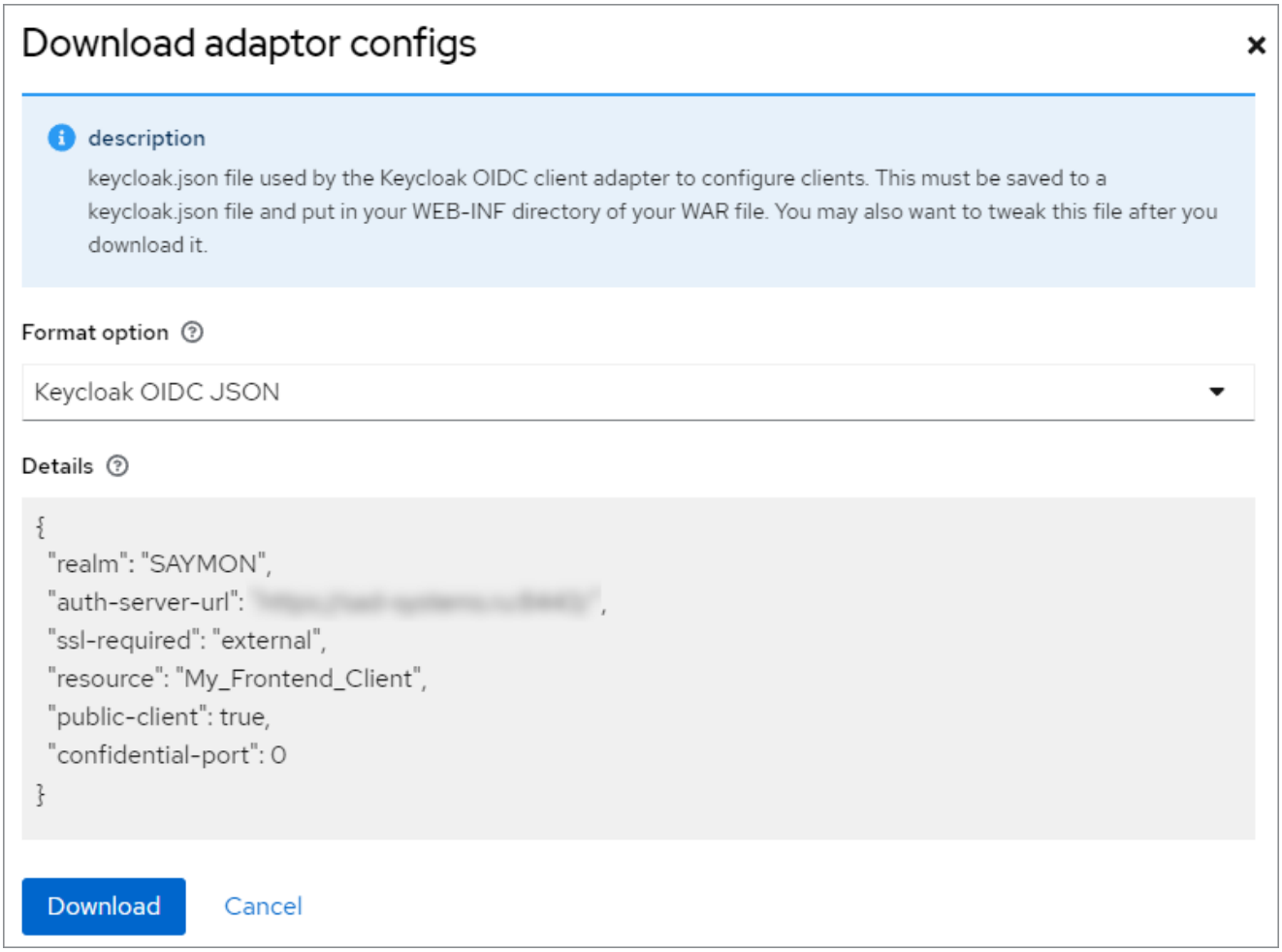


Рис. 29. Файл настроек соединения для Frontend-клиента

Конфигурация web-интерфейса Центрального Пульта

Для активации формы авторизации через Keycloak в web-интерфейсе системы необходимо:

1. Добавить на сервере Центрального Пульта раздел **Authentication** в конфигурационный файл клиента `/etc/saymon/saymon-client.yaml`:

```
authentication:
  availableMethods:
    - "native"
    - "keycloak"
  methodByDefault: "keycloak"
    services:
      keycloak:
        configUri: keycloak_frontend.json
```

   где:

   - **availableMethods** - список доступных для пользователей методов аутентификации;
   - **methodByDefault** - метод аутентификации, который будет предложен

пользователю сразу при отображении аутентификационной формы;

- **configUri** - файл настроек соединения, сгенерированный сервером Keycloak для Frontend-клиента.

WARNING	Файл настроек соединения для Frontend-клиента необходимо поместить в каталог `/etc/saymon`.

2. Обновить страницу Центрального Пульта в web-браузере.

3.6. Управление логированием

Для хранения и дальнейшего анализа изменений в системной работе агентаЦентральный Пульт использует логирование.

Логи агента хранятся в папках: * Linux, Wiren Board 6 - /var/log/saymon; * macOS - /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 илипроцесса) и признак сессии, если событие было инициировано пользователем.

3.6.1. Конфигурация log-файлов агента

Конфигурация log-файлов агента выполняется в файле:

- `<PROGRAMFILES>/SAYMON Agent/conf/logback-winservice.xml` (Windows).
- `/opt/saymon-agent/conf/logback-daemon.xml` (Linux/Mac OS).

Логирование выполняется на уровне INFO. При необходимости возможно дополнительно настроить debug-режим.

Агент выполняет ротацию логов раз в день, старые логи архивируются.

NOTE	Архивация происходит в момент поступления в лог новой информации от агента. Если новых записей нет, архивация не выполняется.

3.6.1.1. Базовое логирование

Пример секции настроек базового логирования:

```
<appender name="FILE-INFO" class="RollingFileAppender">
    <file>log/saymon-agent.info.log</file>
    <rollingPolicy class="TimeBasedRollingPolicy">
        <!-- Daily rollover -->
        <fileNamePattern>log/saymon-agent.info.%d{yyyy-MM-
dd}.log.zip</fileNamePattern>
        <!-- Keep 10 days' worth of history -->
        <maxHistory>10</maxHistory>
        <!-- Limit total size of all archive logs to 50 MB. -->
        <totalSizeCap>50MB</totalSizeCap>
    </rollingPolicy>
    <encoder class="PatternLayoutEncoder">
        <pattern>%d{dd.MM.yyyy HH:mm:ss.SSS} [%-15thread] %-5level %logger{36}
- %msg %kvp%n%rootException</pattern>
    </encoder>
</appender>
```

где

- **<file>log/saymon-agent.info.log</file>** — размещение log-файла;
- **<fileNamePattern>… .zip</fileNamePattern>** — указание на архивацию файлов в формат **.zip**;
- **<maxHistory>10</maxHistory>** — длительность хранения файлов в днях;
- **<totalSizeCap>50MB</totalSizeCap>** — ограничение дискового пространства для хранения логов.

NOTE	Полная информация о настройках доступна в документации по Logback.

3.6.1.2. Активация debug-режима

Для активации debug-режима необходимо:

1. Заменить содержимое файла конфигурации log-файлов агента содержимым файла
   - `logback-winservice-debug.xml` (Windows);
   - `logback-daemon-debug.xml` (Linux/Mac OS).

   В этом файле находится секция настроек debug-режима:

   ```
   <appender name="FILE-DEBUG" class="RollingFileAppender">
   ...
   </appender>
   ```

NOTE	Файл с секцией настроек debug-режима находится в той же папке, что и файл конфигурации log-файлов агента.

2. Перезапустить агента:
    - служба `SaymonAgent` (Windows);
    - служба `saymon-agent` (Linux/Mac OS).

Параметры и структура секции debug-режима аналогичны секции базового логирования.

3.6.2. Конфигурация log-файлов

Уровень логирования задаётся в файле

`/etc/saymon/logger.json`

Конфигурация ротации log-файлов сервера выполняется в файле

`/etc/logrotate.d/saymon`

где

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, урезая его после каждого шага ротации.

3.6.3. 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
```

3.6.4. Назначение ответственного за событие

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

1. Через Web UI:
   a. В панели режимов отображения открыть Журнал событий кнопкой .
   b. Нажать правой кнопкой мыши на требуемое событие, затем выбрать пункт "Назначить ответственного".
   c. Выбрать ответственного за событие из выпадающего списка.

      Имя ответственного пользователя отобразится в соответствующем столбце (Рис. 30):

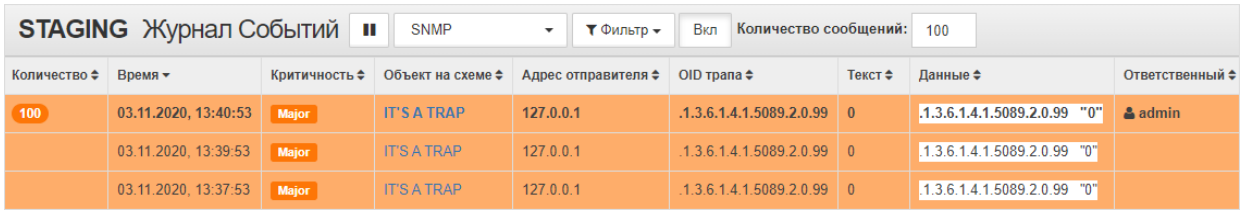


Рис. 30. Журнал событий

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": "..."}'
   ```

3.6.5. Установка ограничения для логирования

Центральный Пульт позволяет установить максимальный объём хранилища двумя

способами:

1. Через Web UI:
    a. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
    b. Перейти в раздел "Журнал событий" (Рис. 31):

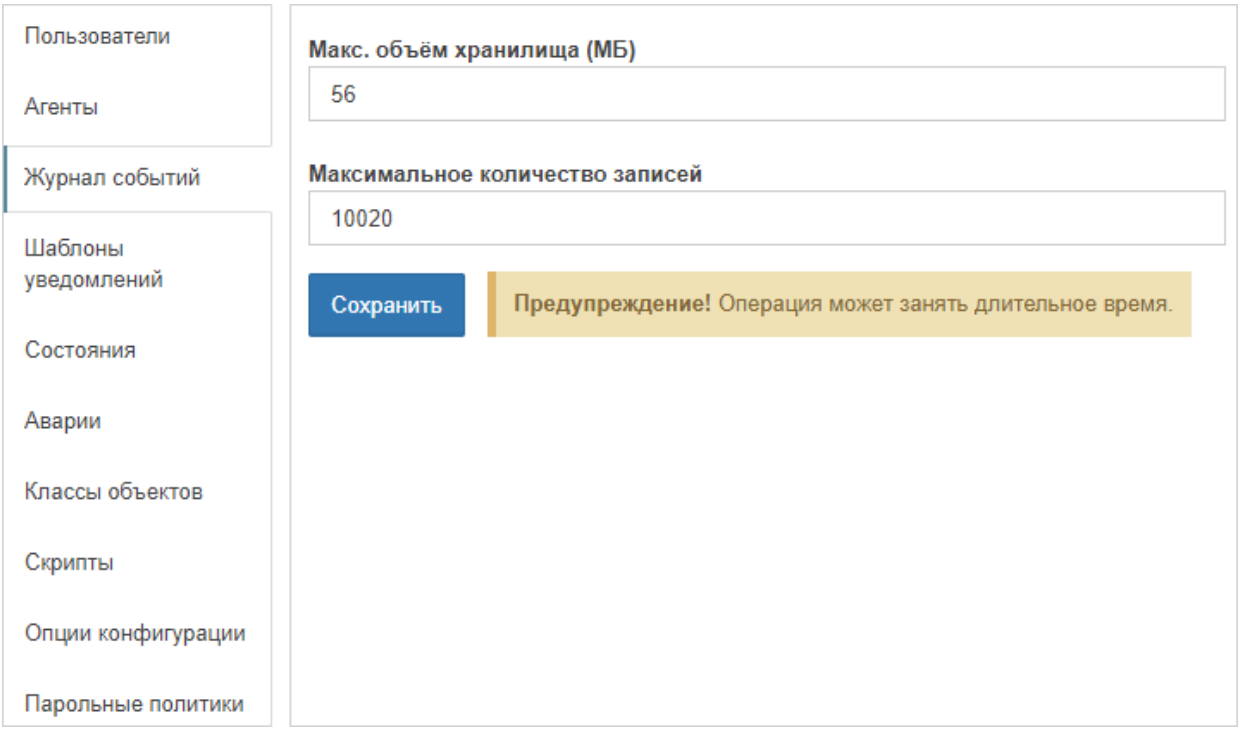


Рис. 31. Журнал событий

    c. Заполнить поля требуемыми значениями.
    d. 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}
```

3.6.6. 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
```

```
}
```

3.6.7. 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
```

3.7. Управление учётными записями пользователей

Раздел содержит информацию об администрировании пользовательских учётных записей.

Группы пользователей позволяют упростить настройку и редактирование прав пользователей. При необходимости настроить нескольким пользователям идентичные

- права на операции,
- доступы к объектам,
- права на просмотр событий в Журнале событий,

достаточно создать группу пользователей, настроить ей необходимые права и добавить в эту группу нужных пользователей. В дальнейшем возможно редактировать права группы. Пользователь может принадлежать к неограниченному числу групп.

Права пользователя и его групп суммируются. Чтобы пользователь мог выполнять те или иные действия в системе, нужно, чтобы соответствующие права были либо у самого пользователя, либо у какой-то из его групп.

3.7.1. Авторизация пользователей через LDAP

Центральный Пульт поддерживает аутентификацию пользователей с помощью сервера

службы каталогов по протоколу LDAP (Active Directory, OpenLDAP, и т.д.).

3.7.1.1. Настройка LDAP

По умолчанию, аутентификацию через LDAP отключена. Для включения этого функционала, в конфигурационный файл сервера нужно добавить раздел `ldap`:

```
"ldap" : {
    "url" : "ldap://example.com:389",
    "suffix" : "dc=example,dc=com",
    "login" : "cn=users,dc=example,dc=com",
    "pass" : "password",
    "allow_self_signed": true,
    "create_user_for_existing_group_only": true,
    "import_non_existing_groups": false,
    "update_existing_groups": false,
}
```

Поведение системы при импорте групп и пользователей настраивается в следующих параметрах:

- `ldap.create_user_for_existing_group_only` – Новый пользователь может авторизоваться через LDAP только если он состоит в группе на LDAP сервере, которая уже есть в Центральном Пульте.
- `ldap.import_non_existing_groups` – При успешном входе LDAP-пользователя, Центральный Пульт импортирует все группы этого пользователя с LDAP-сервера. Группы, которые уже были импортированы не будут затронуты.
- `ldap.update_existing_groups` – При успешном входе LDAP-пользователя, система конвертирует группы пользователей, созданные в Центральном Пульте, до LDAP-групп при совпадении имен групп.

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

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

Учётная запись LDAP сервера, которая указывается в конфигурации, должна иметь административные права. Эта учётная запись должна иметь доступ к пользователям и группам, которые должны быть импортированы в Центральный Пульт.

3.7.1.2. Аутентификация в Центральном Пульте через LDAP

Пользователи и группы, импортированные из LDAP-сервера имеют свои особенности.

У LDAP-пользователей на вкладке Общие отображается поле **Источник** со значением **LDAP**, в списке пользователей они отмечаются иконкой .

LDAP-группы (группы, созданные при авторизации LDAP-пользователей) обозначаются в списке иконкой . Для таких групп недоступна смена имени, в качестве источника указан LDAP.

NOTE	В LDAP-группы можно добавлять не только LDAP-пользователей, но и пользователей из других источников.

Импорт LDAP-пользователей

При первом входе пользователя, аутентифицированного через LDAP, для него автоматически создаётся учётная запись в системе. Синхронизация информации о пользователе происходит при каждом успешном входе этого пользователя.

CAUTION	Если логин пользователя, аутентифицирующегося через LDAP, совпадает с логином другого пользователя, уже зарегистрированного в системе, LDAP-пользователь войти в систему не сможет.

Созданные LDAP-пользователи обладают теми же правами, что и все новые пользователи. Эти права указываются в параметре `user.template.permissions` серверной конфигурации.

Для LDAP-пользователей недоступна вкладка Смена пароля, пароли таких пользователей находятся на LDAP сервере.

Пользователям из LDAP, у которых ещё нет учётной записи в Центральном Пульте, можно ограничить вход в систему, подготовив группы для нужных пользователей. Чтобы учётная запись создавалась только для пользователей, у которых имя группы на LDAP-сервере совпадает с именем существующей (созданной ранее) группы в Центральном Пульте, в конфигурационном файле сервера необходимо указать параметр:

```
{
  ...
  "ldap": {
     ...
     "create_user_for_existing_group_only": true,
     ...
  }
}
```

WARNING	Нельзя использовать вместе с параметром `import_non_existing_groups`. Если у обоих параметров значение `true`, то сервер сначала импортирует группу нового пользователя, а потом авторизует его, так как он входит в существующую группу.

Импорт LDAP-групп

Импорт новых групп

Во время авторизации LDAP-пользователя, Центральный Пульт может импортировать группы этого пользователя из LDAP сервера и автоматически добавить авторизовавшегося пользователя в импортированные группы. Для этого, в конфигурационном файле сервера необходимо указать следующий параметр:

```
{
  ...
  "ldap": {
     ...
     "import_non_existing_groups": true,
     ...
     }
}
```

Обновление существующих групп

Группы, созданные в Центральном Пульте, имена которых совпадают с именами групп из LDAP, можно обновить до LDAP-групп. В конфигурационном файле сервера для этого необходимо указать параметр:

```
{
  ...
  "ldap": {
     ...
     "update_existing_groups": true,
     ...
     }
}
```

WARNING	Если пользователь был исключён из группы в Центральном Пульте - аналога своей группы в LDAP, при следующем входе в систему пользователь автоматически будет возвращён в эту группу.

Удаление LDAP-пользователей

После удаления учётной записи пользователя LDAP с сервера системы, пользователь по-прежнему сможет войти в систему под своими учётными данными, авторизовавшись через LDAP. Учётная запись при этом будет создана заново с правами по умолчанию; права полученные через группы, останутся прежними.

TIP	Параметр **ldap.create_user_for_existing_group_only** не влияет на существующих пользователей. Если опция активирована, а имена групп не совпадают с

	именами групп удалённого пользователя в LDAP, пользователь не сможет создать учётную запись заново.

Чтобы заблокировать LDAP-пользователю доступ к данным, необходимо сменить его статус на **Заблокирован**.

3.7.2. Авторизация пользователей через Keycloak

Для авторизации пользователей с помощью внешнего Keycloak-сервера необходимо добавить:

- раздел **Keycloak** в конфигурационный файл сервера `/etc/saymon/saymon-server.conf`,
- раздел **Authentication** в конфигурационный файл клиента `/etc/saymon/saymon-client.yaml`.

При входе в систему будет отображаться выпадающий список **Метод входа**, где необходимо выбрать **Keycloak** и нажать кнопку Войти. Пользователя перенаправит на Keycloak-сервер, где необходимо ввести логин и пароль от учётной записи Keycloak.

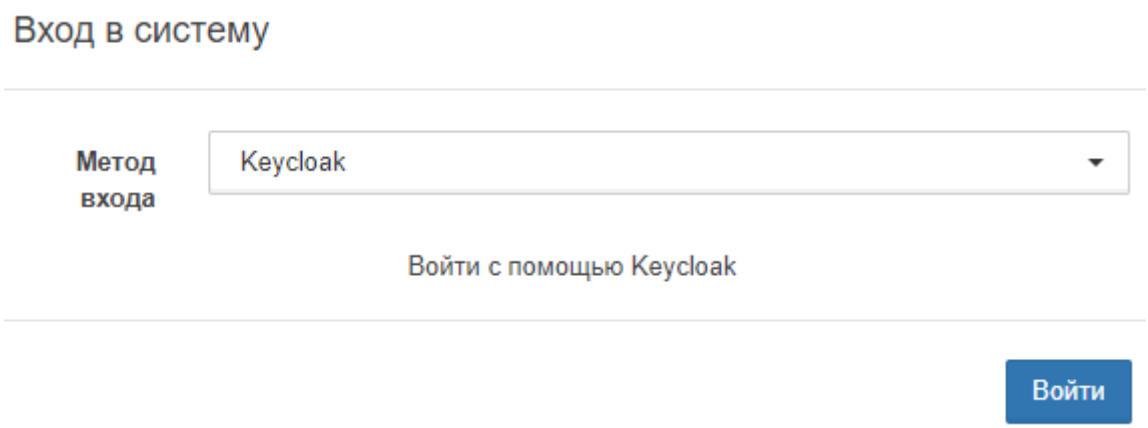


Рис. 32. Окно выбора метода входа

3.7.2.1. Учётные записи Keycloak-пользователей

При первом входе пользователя, аутентифицированного через Keycloak, для него автоматически создаётся учётная запись в системе. У таких пользователей на вкладке Общие отображается поле **Источник** со значением **Keycloak**, в списке пользователей они отмечаются иконкой .

CAUTION	Если логин пользователя, аутентифицирующегося через Keycloak, совпадает с логином другого пользователя, уже зарегистрированного в системе, Keycloak-пользователь войти в систему не сможет.

По умолчанию такие пользователи обладают теми же правами, что и все новые пользователи.

Для Keycloak-пользователей недоступна вкладка Смена пароля.

3.7.2.2. Удаление Keycloak-пользователей

После удаления учётной записи пользователя Keycloak с сервера системы, пользователь по-прежнему сможет войти в систему под своими учётными данными, авторизовавшись через Keycloak. Учётная запись при этом будет создана заново с правами по умолчанию.

Чтобы заблокировать Keycloak-пользователю доступ к данным, необходимо сменить его статус на **Заблокирован**.

3.7.3. Создание учётной записи

При создании пользователей и групп в именах и паролях допускаются любые символы, кроме символа @.

Создание учётных записей пользователей осуществляется двумя способами:

1. Через Web UI:
    a. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
    b. Перейти в раздел "Журнал событий" (Рис. 33):

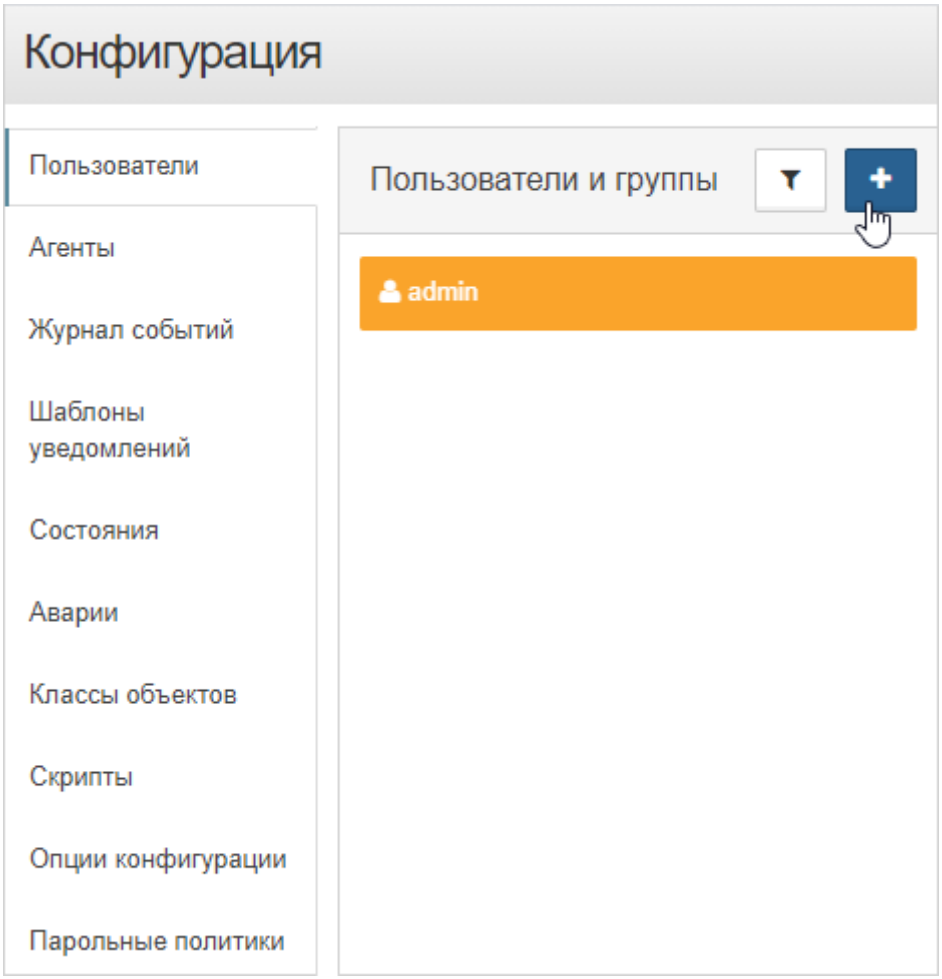


Рис. 33. Журнал событий

c. Нажать кнопку  и выбрать добавление пользователя.

d. Ввести логин пользователя, пароль и подтверждение.

e. 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
```

3.7.4. Назначение пользователям прав доступа

NOTE	*Список операций с подробным описанием содержится в Приложении А на стр. 173.*

Настройка прав пользователей осуществляется двумя способами:

1. Через Web UI:

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

   b. Перейти в раздел "Пользователи".

   c. Выбрать нужного пользователя или группу из списка.

   d. Открыть вкладку "Права на операции" (Рис. 34):

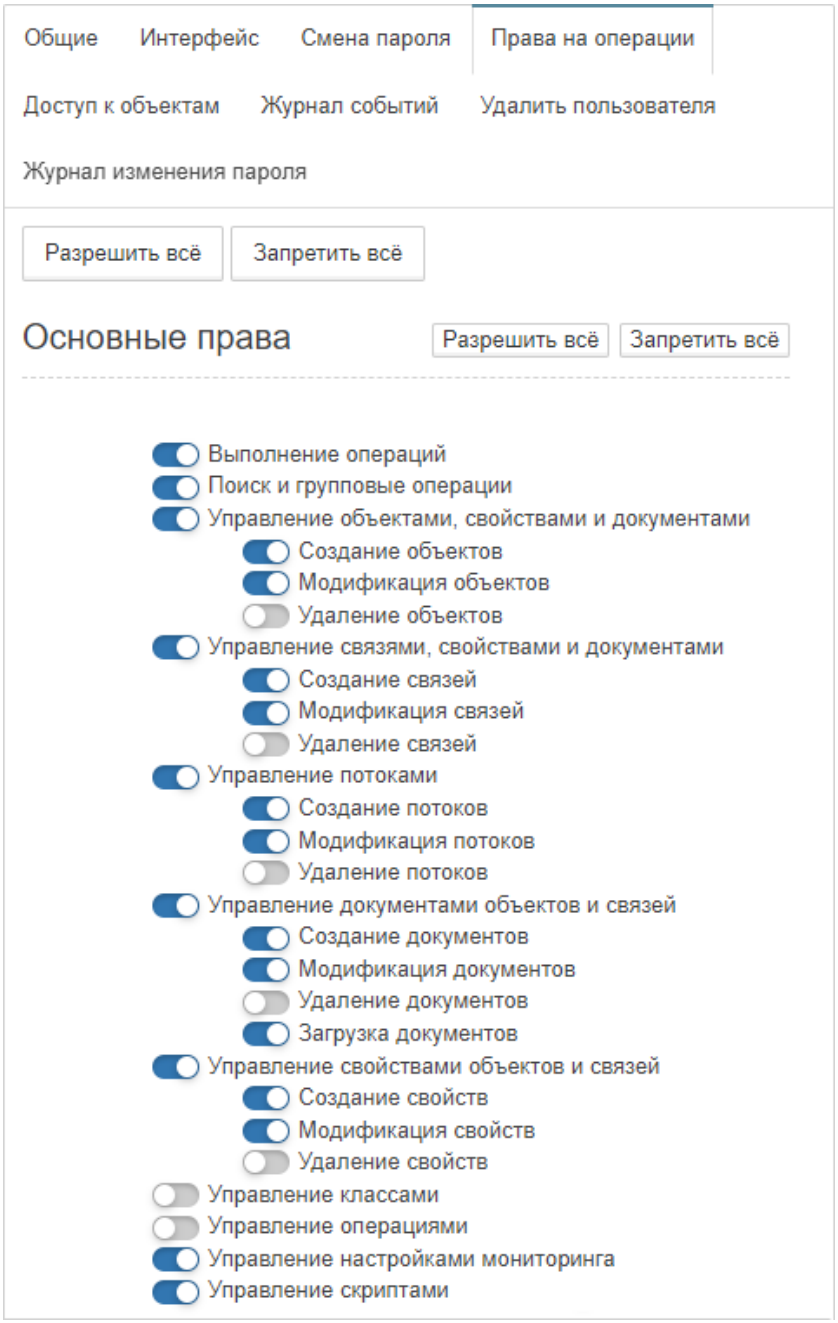


Рис. 34. Права на операции

e. Отредактировать права пользователя или группы.

WARNING	*Если пользователь принадлежит к группам, права пользователя и его групп суммируются.*

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
```

3.7.5. Назначение парольных политик

Настройка парольных политик осуществляется через Web UI.

1. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
2. Перейти в раздел "Парольные политики" (Рис. 35):

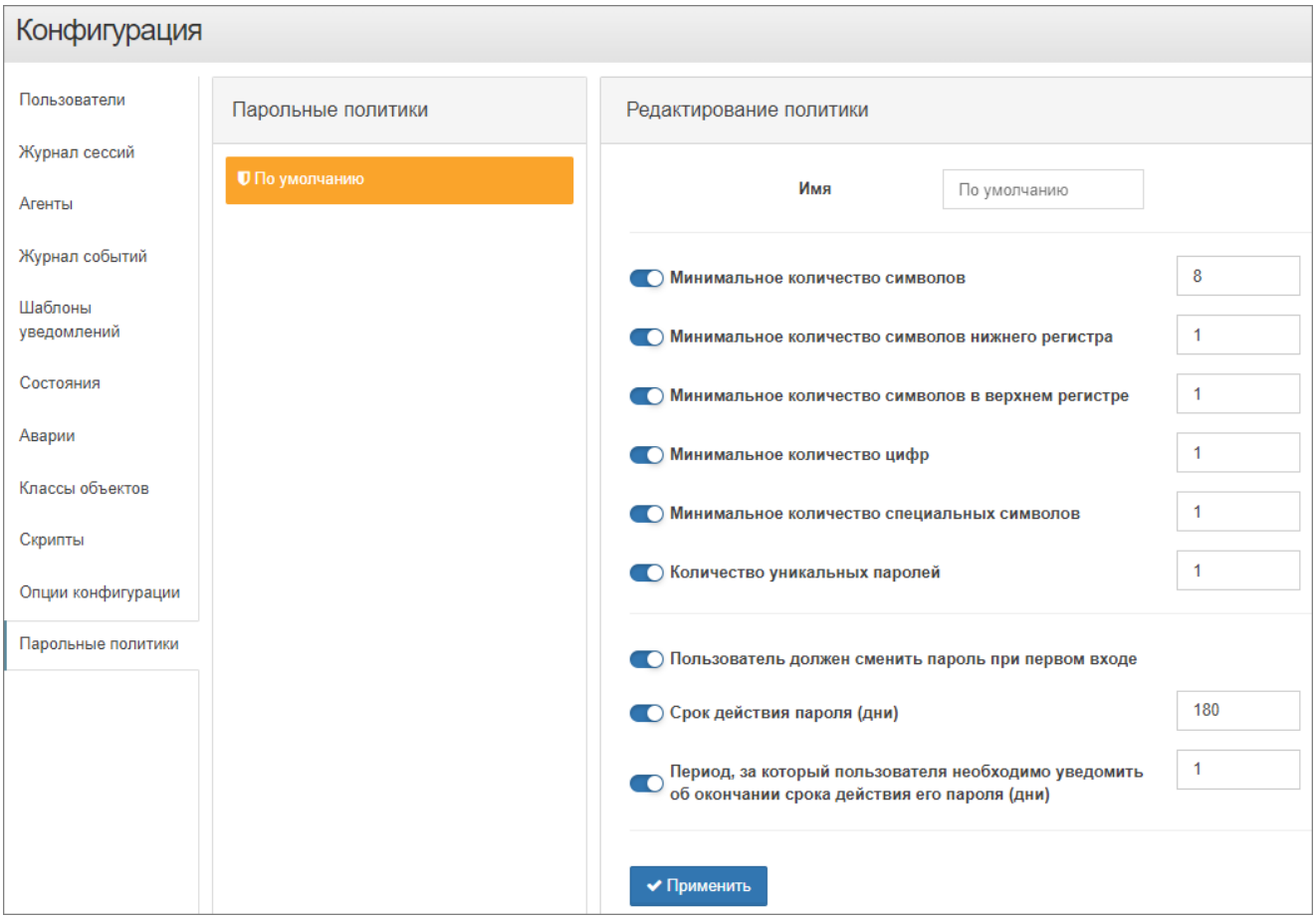


Рис. 35. Парольные политики

3. Установить необходимые требования к паролям пользователей.

4. Нажать кнопку Применить.

3.7.6. Изменение пароля от учётной записи

Смена пароля пользователей осуществляется двумя способами:

1. Через Web UI:
   a. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
   b. Перейти в раздел "Пользователи".
   c. Выбрать нужного пользователя из списка.
   d. Открыть вкладку "Смена пароля" (Рис. 36):

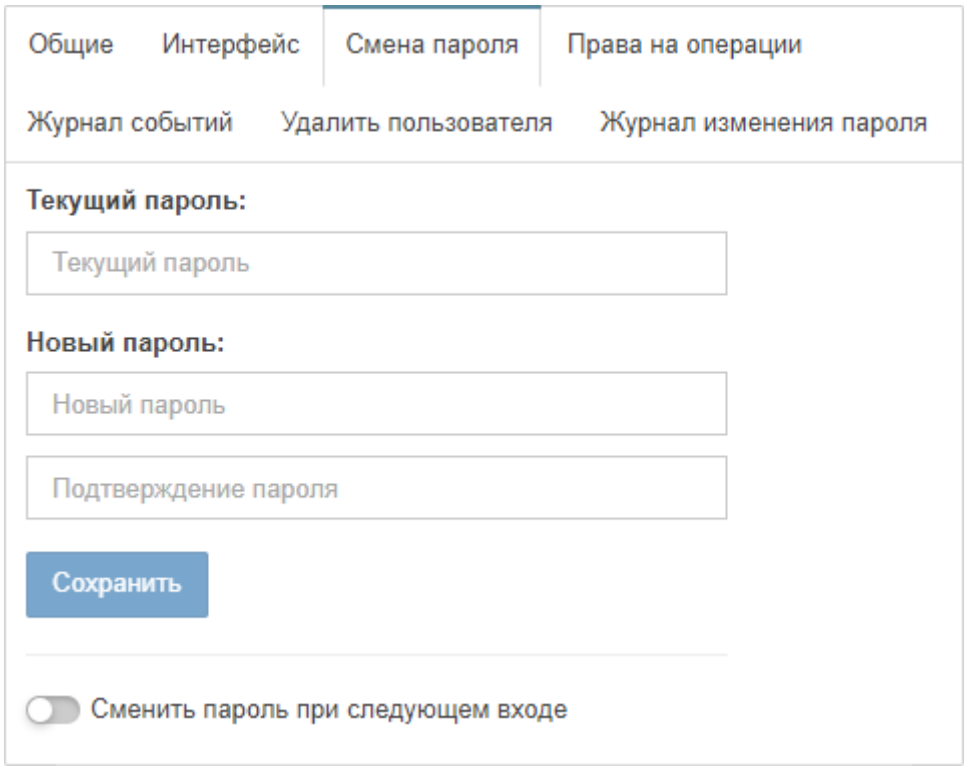


Рис. 36. Смена пароля пользователя

   e. Ввести новый пароль пользователя и подтверждение.

WARNING	*При смене собственного пароля необходимо также ввести текущий пароль.*

   f. Нажать кнопку Сохранить.

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
```

3.7.7. Настройка доступа к объектам

Настройка доступа к объектам осуществляется через Web UI.

1. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
2. Перейти в раздел "Пользователи".
3. Выбрать нужного пользователя или группу из списка.
4. Открыть вкладку "Доступ к объектам" (Рис. 37):

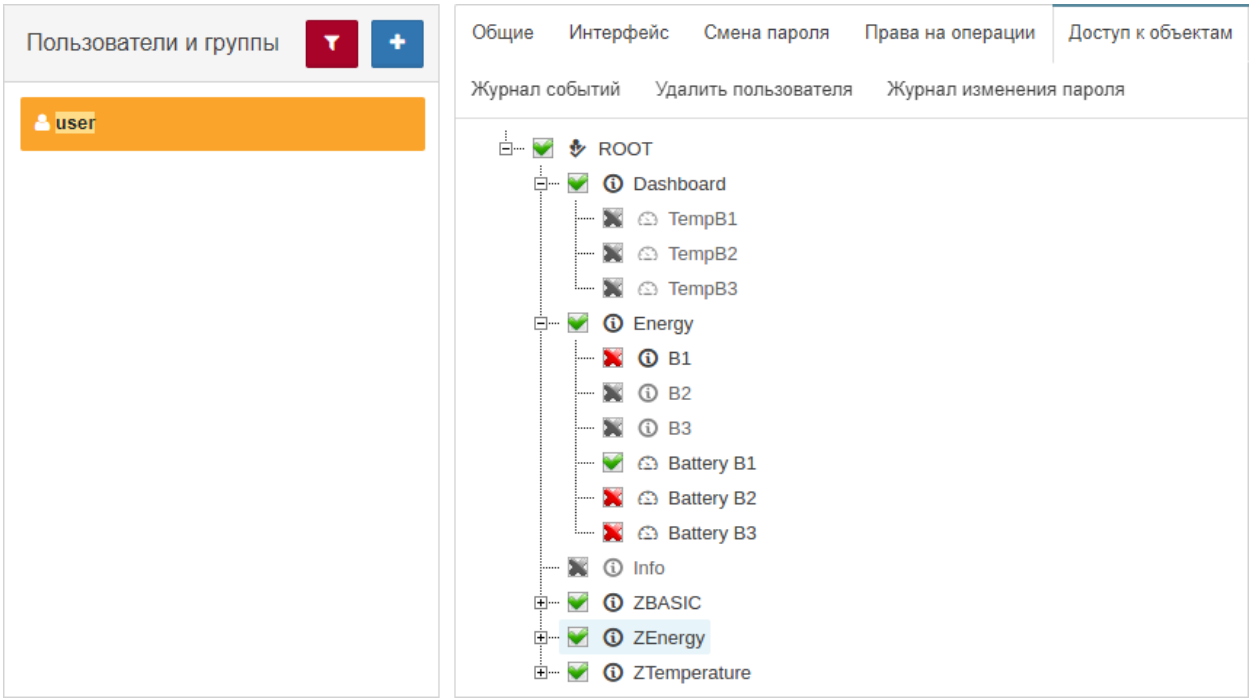


Рис. 37. Доступ к объектам

5. Отметить объекты, к которым необходимо предоставить/заблокировать доступ.

WARNING	*Объекты, недоступные для группы, становятся недоступными для всех членов группы.*

3.7.8. Блокировка учётной записи

Блокировка учётных записей пользователей осуществляется двумя способами:

1. Через Web UI:
    a. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
    b. Перейти в раздел "Пользователи".
    c. Выбрать нужного пользователя из списка.
    d. Открыть вкладку "Общие" (Рис. 38):

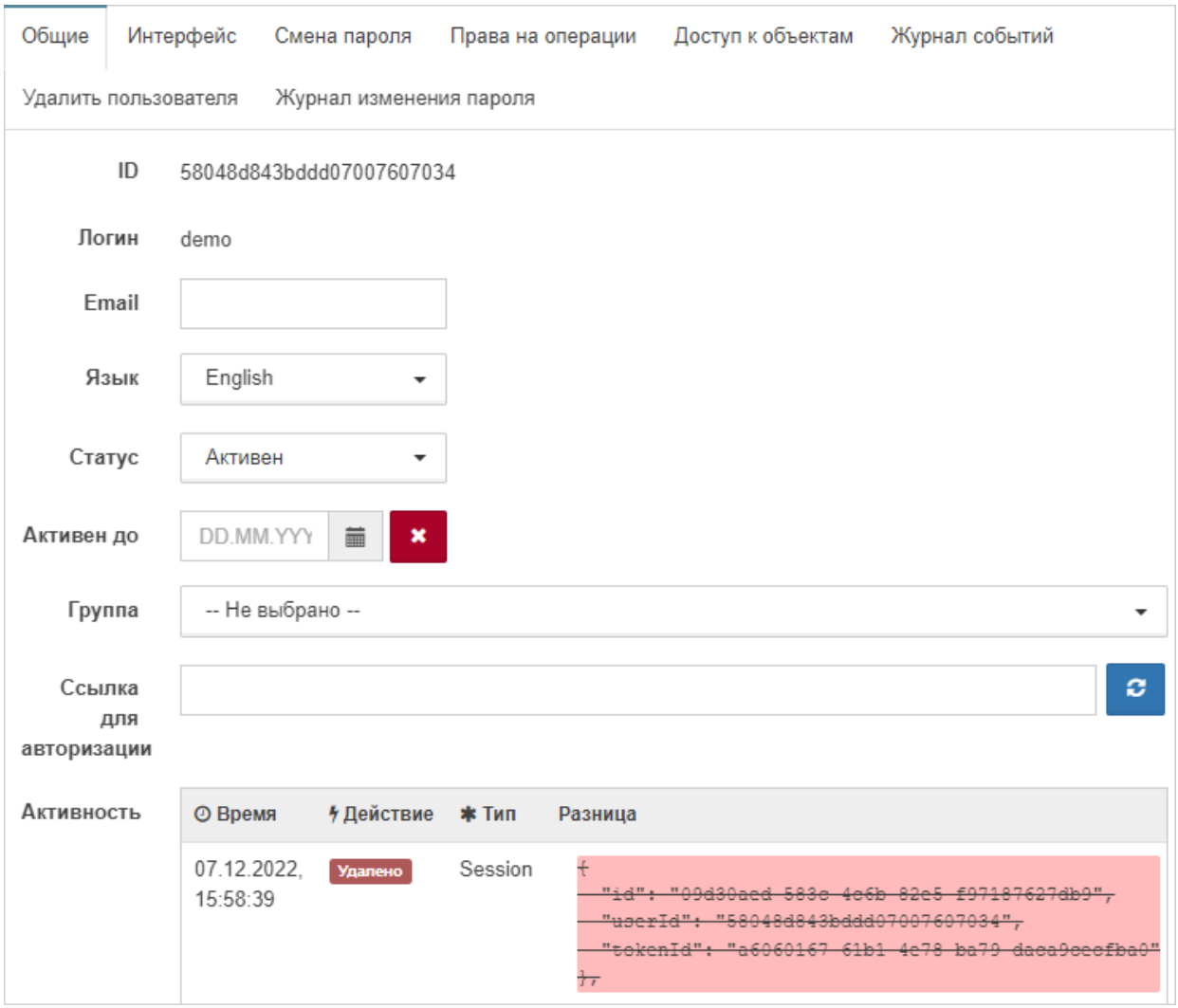


Рис. 38. Общие параметры пользователя

    e. В выпадающем списке "Статус" выбрать "Заблокирован".

    В поле **Активен до** можно задать дату и время блокировки учётной записи. При наступлении указанного времени статус учётной записи будет автоматически изменён на **Заблокирован**.

WARNING	*В случае если указанное время меньше текущего, статус учётной записи немедленно изменяется на **Заблокирован**.*

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/password

curl -X PATCH $url -u $login:$password -H "Content-Type: application/json" \
-d @- <<EOF
{
"status": "3",
}
EOF
```

3.7.9. Удаление учётной записи

Удаление учётных записей осуществляется двумя способами:

1. Через Web UI:
   a. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
   b. Перейти в раздел "Пользователи".
   c. Выбрать нужного пользователя из списка.
   d. Открыть вкладку "Удалить пользователя" (Рис. 39):

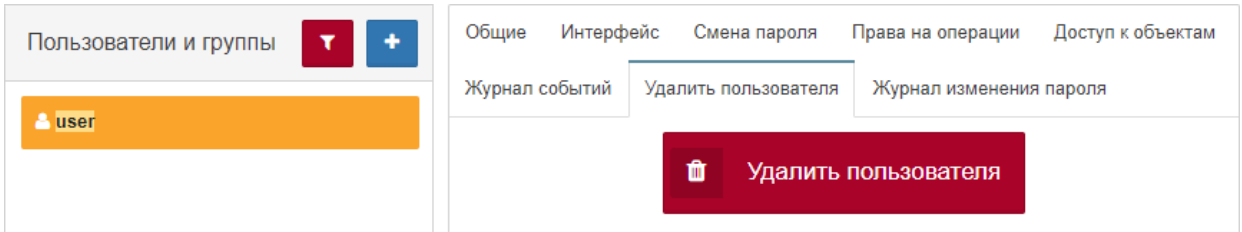


Рис. 39. Удаление пользователя

   e. Нажать кнопку удаления пользователя.
   f. 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
```

WARNING	*После удаления учётной записи пользователя LDAP/Keycloak с сервера системы пользователь по-прежнему сможет войти в систему под своими учётными данными, авторизовавшись через LDAP/Keycloak. Учётная запись при этом будет создана заново с правами по умолчанию; права, полученные через группы, останутся прежними.* *Чтобы заблокировать такому пользователю доступ к данным, достаточно сменить его статус на **Заблокирован**.*

3.7.9.1. Просмотр журнала действий пользователей

Просмотр журнала действий пользователей осуществляется двумя способами:

1. Через Web UI:
    a. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
    b. Перейти в раздел "Пользователи".
    c. Выбрать нужного пользователя из списка.
    d. Открыть вкладку "Журнал активности" (Рис. 40):

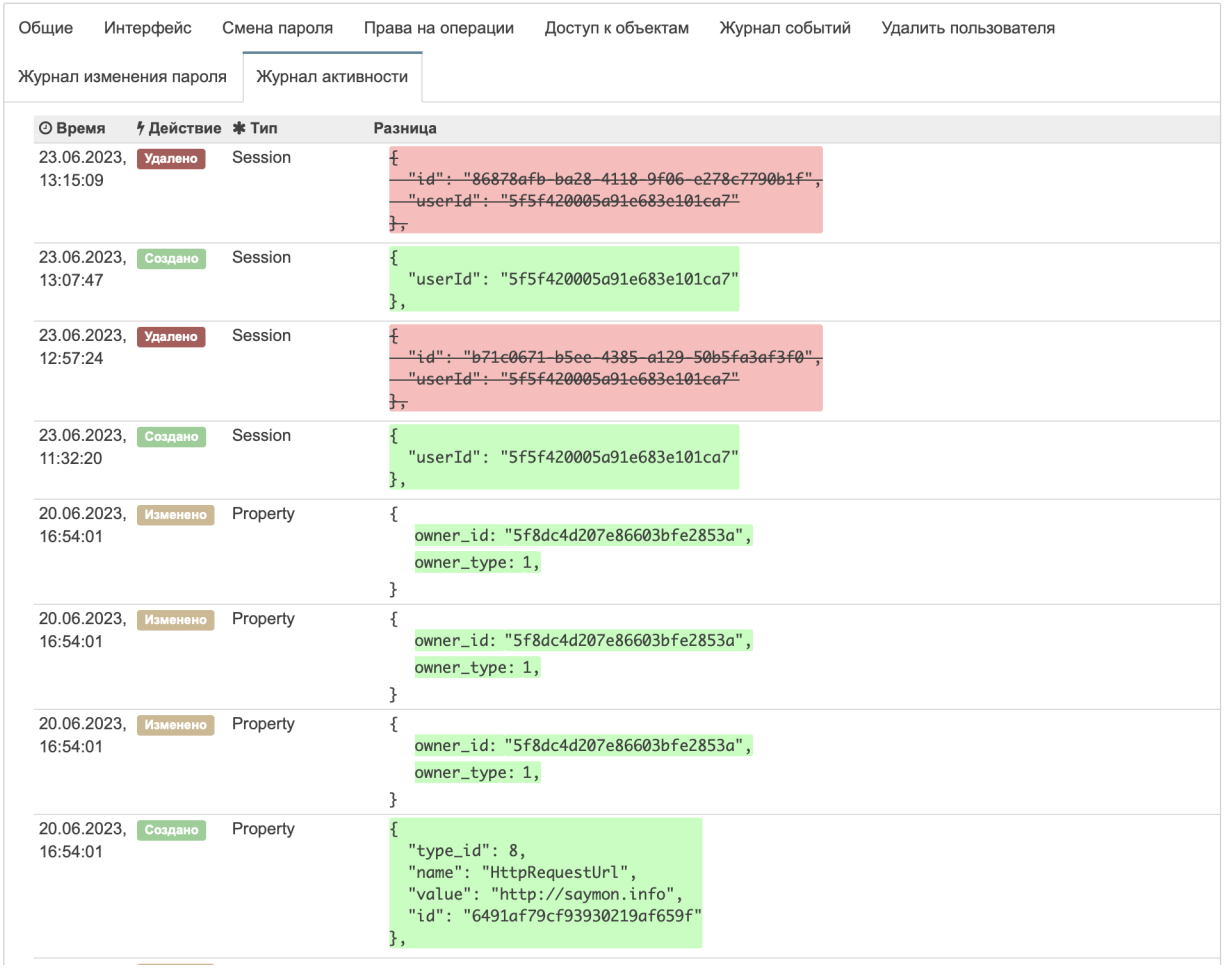


Рис. 40. Журнал активности

e. В поле "Активность" указаны все действия выбранного пользователя в системе.

2. Через REST API:

```
GET /node/api/users/:id/audit-log
```

Пример (bash):

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

curl -X GET $url -u $login:$password \
-G --data-urlencode "limit=5"
```

3.8. 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.

3.8.1. Создание объекта

Создать объект можно двумя способами:

1. Через Web UI:
   a. Нажать кнопку + Создать объект на панели инструментов.
   b. В окне "Новый объект" ввести имя объекта и выбрать класс объекта (Рис. 41):

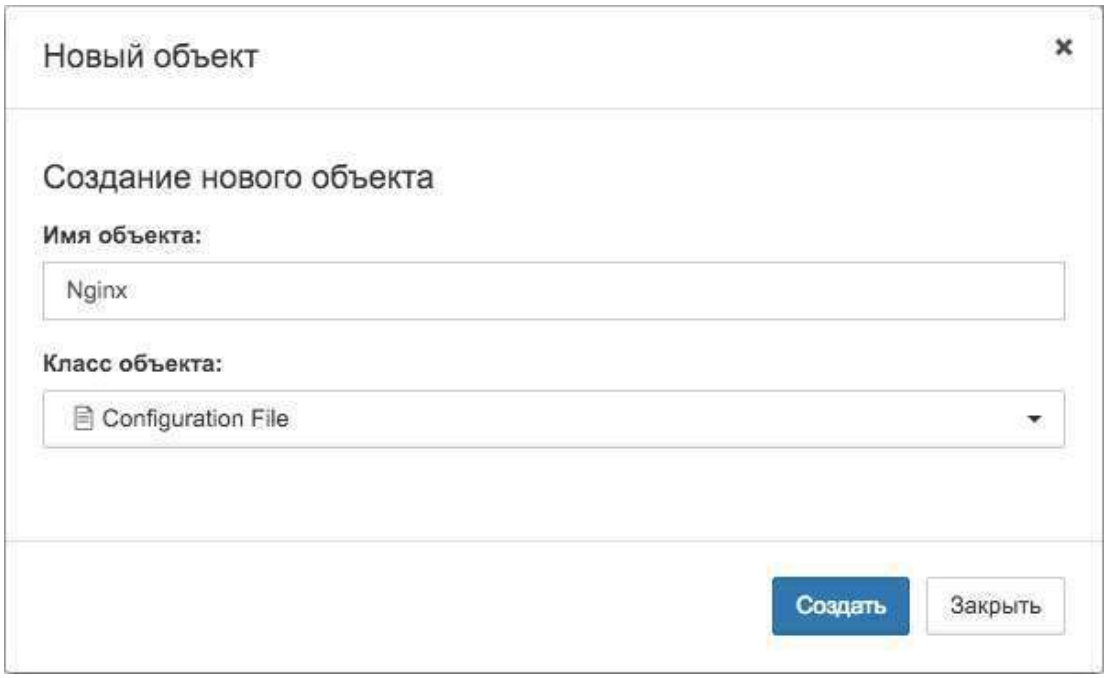


Рис. 41. Окно создания нового объекта

   c. Нажать кнопку Создать.
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
```

WARNING	*"parent_id" - ID родительского объекта для создаваемого.*

3.8.2. Клонирование объекта

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

Клонировать объект можно двумя способами:

1. Через Web UI:
    a. Вызвать контекстное меню клонируемого объекта щелчком правой кнопкой мыши.
    b. 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
    ```

3.8.3. Удаление объекта

Удалить объект можно тремя способами:

1. Через Web UI (контекстное меню объекта):
    a. Вызвать контекстное меню удаляемого объекта щелчком правой кнопкой мыши.
    b. Выбрать пункт меню "Удалить".
    c. Подтвердить удаление объекта во всплывающем окне.
2. Через Web UI (режим удаления элементов):

a. Нажать кнопку  на панели хлебных крошек.

b. Нажать на такую же иконку на удаляемом объекте.

c. 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
```

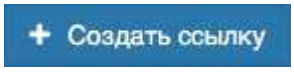
3.8.4. Создание ссылки на объект

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

С помощью ссылок можно создавать связи и потоки между объектами, которые находятся в разных частях инфраструктуры.

Создать ссылку можно двумя способами:

1. Через Web UI:

a. Нажать кнопку 

на панели инструментов.

b. В появившемся всплывающем окне "Новая ссылка" (Рис. 42) выбрать из выпадающего списка объект, на который создаётся ссылка:

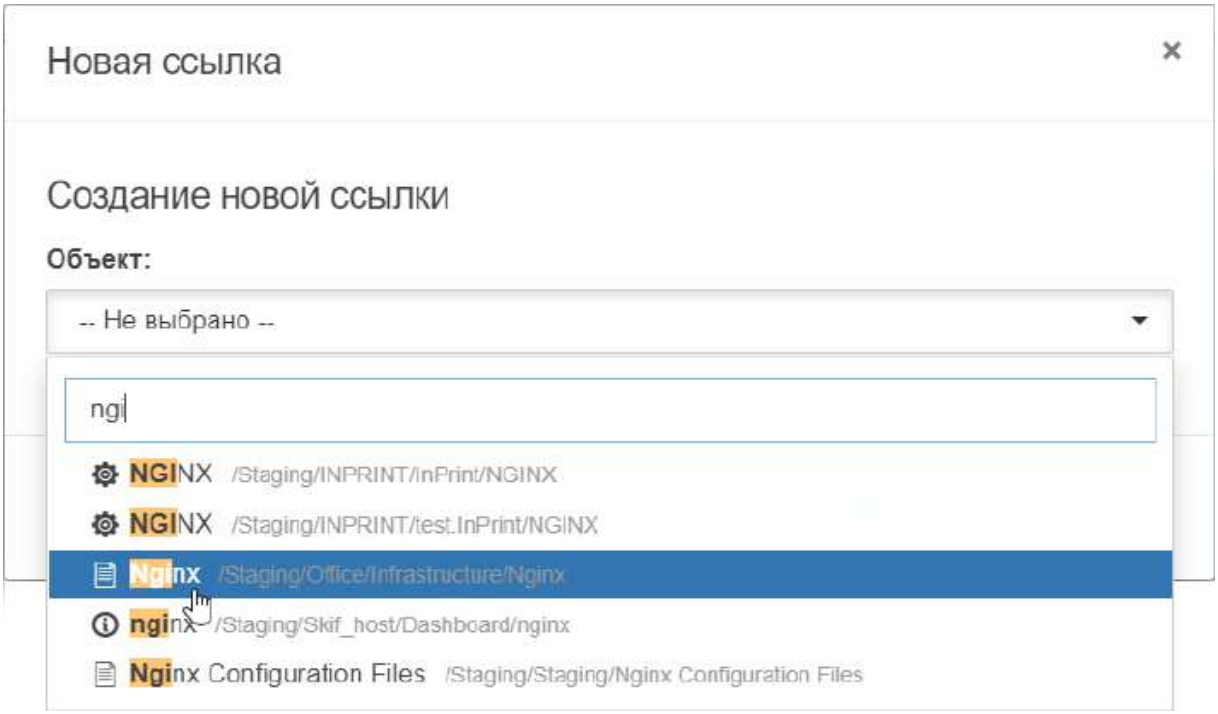


Рис. 42. Окно создания нового объекта

c. Нажать кнопку **Создать**.

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"}
```

3.8.5. Создание связи

Связь может отражать как физическое соединение между объектами (например, соединение сервера с маршрутизатором с помощью Ethernet-кабеля), так и логическое отношение (например, поток данных от одного программного компонента к другому).

Связи между объектами также могут являться объектами мониторинга.

Создать связь можно двумя способами:

1. Через Web UI:

   a. Перейти в режим создания связей, нажав кнопку **+ Создать связь** на панели

инструментов.

b. После того, как на всех объектах появится соответствующий символ , нажать на него на исходном объекте и, удерживая, переместить курсор на целевой объект (Рис. 43):

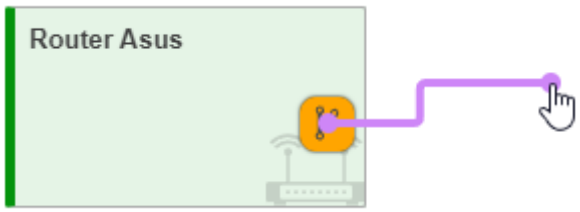


Рис. 43. Создание связи

Созданная связь отобразится между выбранными объектами (Рис. 44):

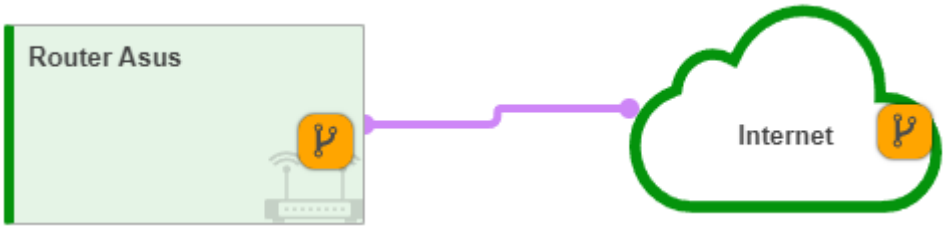


Рис. 44. Связь между объектами

c. 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
```

3.8.6. Удаление связи

Удалить связь можно двумя способами:

1. Через Web UI:
   a. Нажать кнопку  на панели хлебных крошек.
   b. Нажать на такую же иконку на удаляемой связи.
   c. 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
   ```

3.9. Настройка интерфейса

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

3.9.1. 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 **/etc/saymon/saymon-client.yaml**

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 необходимо обновить страницу в браузере.

3.9.2. Настройка заголовка web-интерфейса

Для изменения заголовка в web-интерфейсе:

1. Открыть файл **/etc/saymon/saymon-client.yaml**.
2. В строке "title" ввести желаемое имя:

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

3. Обновить страницу браузера.

3.9.3. 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
}
```

3.9.4. 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;
}
```

3.9.5. 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 Web UI:
    a. В панели инструментов нажать на имя пользователя и выбрать в меню пункт "Конфигурация".
    b. Перейти в раздел "Состояния".
    c. В списке состояний выбрать то, которое требуется изменить.
    d. При необходимости изменить имя в соответствующем поле.
    e. Настроить основной цвет, цвет тени, цвет строки таблицы и фон при помощи цветовой палитры или методом ввода номера цвета.
2. Через файл конфигурации:
    a. Создать файл: **/usr/local/saymon/saymon.local/css/saymon.local.css**.
    b. 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.

3.10. Настройка мониторинга

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

NOTE	*Подробную информацию о каждом типе проверок см. в "Руководстве пользователя" в подразделе 4.2.1 "Настройка процесса мониторинга и типов проверок" на стр. 43.*

Далее рассмотрены несколько примеров процесса мониторинга.

3.10.1. 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.

3.10.2. 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.

3.10.3. 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.

3.10.4. Проверка доступности web-ресурса

Данный тип мониторинга позволяет убедиться не только в работоспособности web-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 web-ресурса необходимо:

1. Установить, сконфигурировать и запустить агента на наблюдаемом ПК или сервере.
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.

3.10.5. Безагентный мониторинг web-сервера

Существует ряд случаев, при которых установка агента на сервере невозможна. В таких случаях рекомендуется 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 web-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 web-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.

3.11. Страницы для внедрения в IFrame

Виджет IFrame, предназначенный для внедрения в интерфейс содержимого сторонних ресурсов, можно также использовать и со специально настроенными информационными страницами внутри Центрального Пульта.

3.11.1. Страница с авариями

В адресной строке страницы аварий возможно указать дополнительные URL-параметры, влияющие на представление панели элементов управления и отображение данных в таблице.

Пример для отключения всех элементов управления, кроме кнопки экспорта данных в Excel, и применения ранее сохранённого фильтра **Major** (вместо **127.0.0.1** указать адрес нужного сервера Центрального Пульта):

```
http://127.0.0.1/incidents.html?header=off&exportSection=on&filterName=Major
```

3.11.1.1. Параметры элементов управления

Параметр	Описание
header	Включает/выключает отображение панели, содержащей элементы управления фильтрацией данных в таблицах; • on - отображение панели включено, • off - отображение панели выключено. По умолчанию - **on** - отображение панели включено. **WARNING**: При выключенном отображении панели элементов управления выключаются все описанные в таблице элементы. В этом случае, при необходимости включить отдельный элемент, нужно отдельно вписать его в адресную строку со значением **on**.
cache	Включает/выключает загрузку сделанных пользователем настроек из кэша браузера: будут ли настройки сохраняться при обновлении страницы браузера; • on - загрузка настроек включена, • off - загрузка настроек выключена.

Параметр	Описание
entityHref	Включает/выключает отображение ссылок на аварийные элементы в столбце **Объект/связь** и их родительские объекты в столбце **Путь**; • on - отображение ссылок включено, • off - отображение ссылок выключено.
exportSection	Включает/выключает отображение кнопки экспорта данных в Excel; • on - отображение кнопки экспорта данных включено, • off - отображение кнопки экспорта данных выключено.
filterSection	Включает/выключает отображение секции расширенного фильтра; • on - отображение секции фильтра включено, • off - отображение секции фильтра выключено.
logo	Включает/выключает отображение логотипа; • on - отображение логотипа включено, • off - отображение логотипа выключено.
presetSection	Включает/выключает отображение секции представлений; • on - отображение секции представлений включено, • off - отображение секции представлений выключено.
propertyFieldsSection	Включает/выключает отображение секции с выбором полей свойств; • on - отображение секции с выбором полей свойств включено, • off - отображение секции с выбором полей свойств выключено.

Параметр	Описание
typeSelect	Включает/выключает отображение селектора выбора типа данных - выбор между активными авариями и историей аварий; • on - отображение селектора включено, • off - отображение селектора выключено.

3.11.1.2. Параметры фильтра

Параметр	Описание
filterName	Имя сохранённого фильтра, который сразу будет применён к отображаемым данным.
timeFilter	Включает/выключает отображение дополнительного фильтра по временному диапазону; • on - временной фильтр включен, • off - временной фильтр выключен. По умолчанию - **off** - временной фильтр выключен. 
type	Выбор типа отображаемых данных - список исторических аварий; • active - отображать список активных аварий, • history - отображать список исторических аварий. По умолчанию отображается последний выбранный тип.

3.11.2. Страница с графиками

В системе возможно настроить специальную страницу с набором графиков из одного или нескольких объектов (Рис. 45):

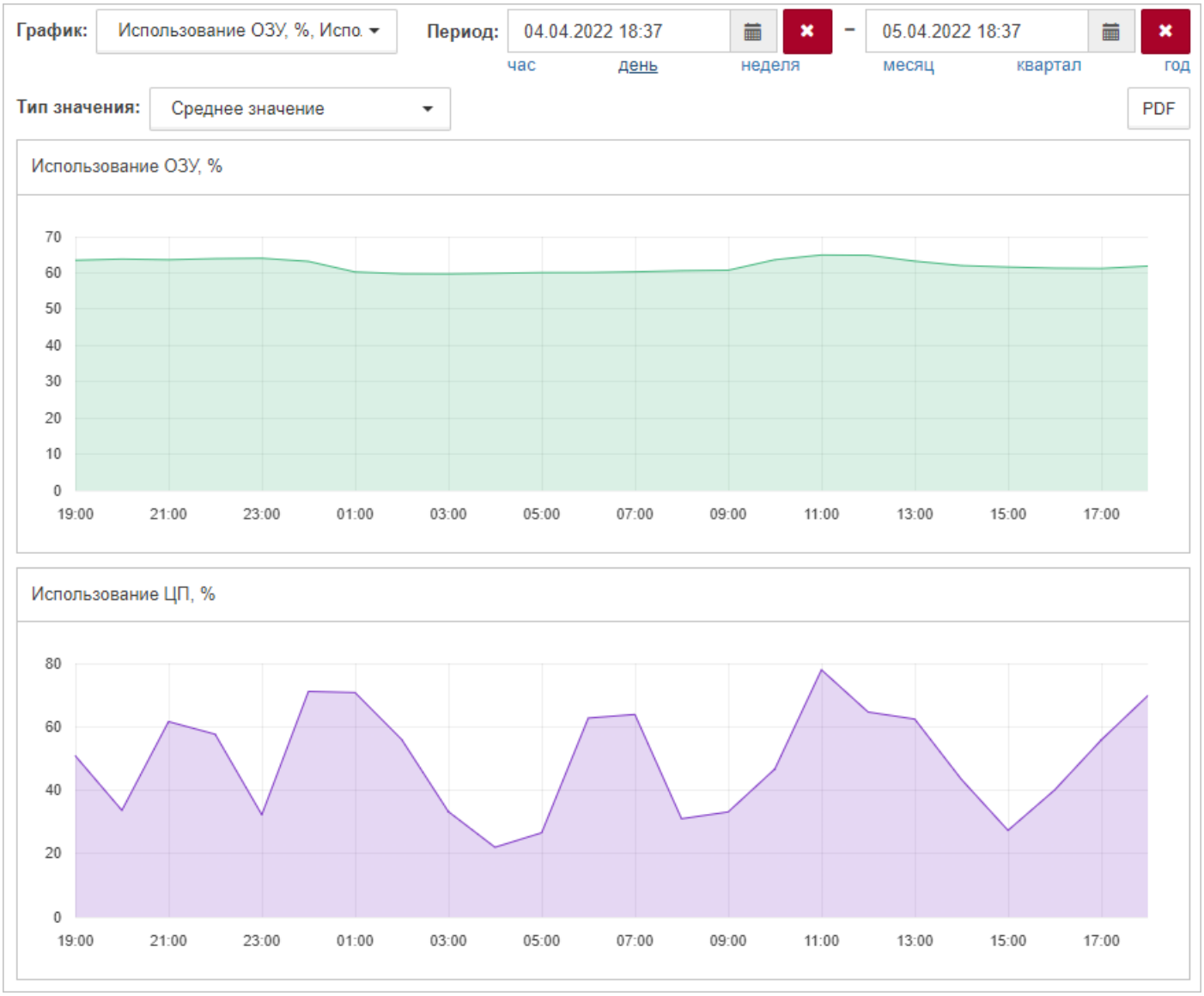


Рис. 45. Пример настроенной страницы графиков

В верхней части страницы представлены элементы управления (Рис. 46):

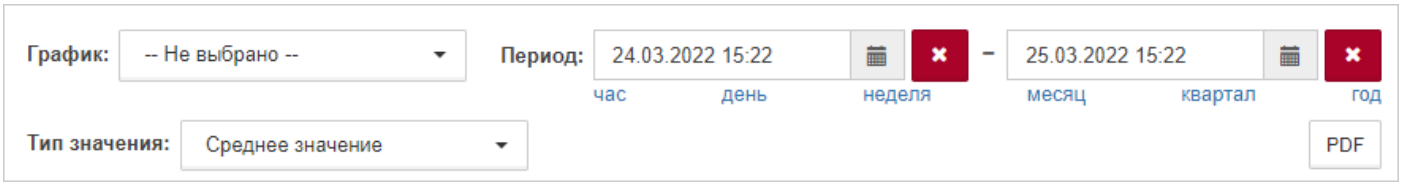


Рис. 46. Элементы управления страницы графиков

Элементы управления страницы графиков:

Элемент	**Описание**
График	Выпадающий список, в котором можно выбрать отображаемые графики подключенных метрик.

Элемент	Описание
Период	Временной период для отображения графиков. При необходимости можно быстро ввести период за последние • час, • день, • неделю, • месяц, • квартал, • год, нажав соответствующую кнопку под полями выбора начала и конца периода.
Тип значения	Если при запросе графика по метрике возвращается большой массив точек, то для оптимального построения графика производится автоматическое объединение точек.Способ вычисления значений точек графиков: • **среднее значение** - значения точек графика рассчитываются как среднее арифметическое значений метрики; • **минимальное значение** - значения точек графиков соответствуют минимальным значениям метрики; • **максимальное значение** - значения точек графиков соответствуют максимальным значениям метрики.
PDF	Кнопка сохранения страницы в PDF-формате.

Страница с графиками доступна по ссылке вида

*http://**<адрес сервера Центрального Пульта>**/iframe-widgets.html?#widget/selectable-chart/**<ID основного объекта>**/**<дополнительные параметры>***

где

- **<адрес сервера Центрального Пульта>** - IP-адрес или FQDN инсталляции Центрального Пульта;
- **<ID основного объекта>** - идентификатор объекта, в свойствах которого хранится список метрик и их отображаемые названия;
- **<дополнительные параметры>** - настройки, которые определяют первоначальный вид страницы при её загрузке.

Пример:

http://**127.0.0.1**/iframe-widgets.html?#widget/selectable-chart/**5fcc9a4de0a5393f45fb99f3**/**chart=61a662bdde9e7970b38b340c**:**percentageUsage.combined**&**chart=TOTAL.percentUsed**&**period=last-24hours**

В данном примере:

- сервер работает на локальной машине и доступен по адресу **127.0.0.1**;
- основным объектом указан объект с идентификатором **5fcc9a4de0a5393f45fb99f3**;
- при загрузке страницы отображаются график метрики **percentageUsage.combined** объекта с идентификатором **61a662bdde9e7970b38b340c** и график метрики **TOTAL.percentUsed** основного объекта за прошедшие сутки.

Возможные метрики для построения графиков необходимо указать свойствах основного объекта.

В имени свойства указывается имя метрики в формате:

chart:<имя метрики>

При необходимости добавить метрику из другого объекта нужно добавить идентификатор объекта:

chart:<ID внешнего объекта>:<имя метрики>

В значении свойства задаётся отображается имя графика:

<желаемое название отображаемого графика>

Примеры:

Имя свойства: chart:TOTAL.percentUsed

Значение свойства: Использование ОЗУ, %

Описание: График метрики ***TOTAL.percentUsed*** основного объекта

Имя свойства: chart:61a662bdde9e7970b38b340c:percentageUsage.combined

Значение свойства: Использование ЦП, %

Описание: График метрики ***percentageUsage.combined*** объекта с ID ***61a662bdde9e7970b38b340c***

Дополнительные параметры страницы графиков:

Параметр	Возможные значения	Описание
aggregate	**aggregate=avg** - выбираются средние значения, **aggregate=max** - выбираются максимальные значения, **aggregate=min** - выбираются минимальные значения.	Тип значения, применяемый при загрузке страницы. По умолчанию применяется функция усреднения.
chart	chart:<имя метрики основного объекта> **chart:packetsTransmitted** chart:<ID нужного объекта>:<имя метрики нужного объекта> **chart:61a662bdde9e7970b38b340c:freeMemory** *Используемые в URL метрики должны быть заранее настроены в свойствах основного объекта.*	Графики, которые будут отображаться при загрузке страницы. Если не указать данный параметр, то по умолчанию откроется страница без графиков.
period	**period=last-hour** - за последний час, **period=last-24hours** - за последний день, **period=last-7days** - за последнюю неделю, **period=last-30days** - за последний месяц, **period=last-90days** - за последний квартал, **period=last-365days** - за последний год.	Период времени для построения графиков, выбираемый при загрузке страницы. По умолчанию отображаются графики за последний час.

Параметр	Возможные значения	Описание
widget-instance-id	Значение параметра - имя представления, уникальное в рамках основного объекта. В качестве значения может быть использован любой набор символов. Примеры: • **widget-instance-id=1** - сохранение настроек в представлении с именем **1**; • **widget-instance-id=example** - сохранение настроек в представлении с именем **example**.	Сохранение настроек страницы в отдельном представлении с уникальным именем. По умолчанию настройки не записываются в представление.

Параметр **widget-instance-id** позволяет сохранять все производимые пользователем настройки в представлении с соответствующим именем.

Имена представлений задаются пользователями произвольно.

TIP	*Функционал будет полезен при необходимости одновременного просмотра графиков одних и тех же метрик с различными настройками отображения.*

Если встроить одну и ту же страницу с графиками в несколько IFrame-виджетов, то все изменения, вносимые через элементы управления на одном из виджетов, будут применяться и к остальным. Изменения с настройками отображения страницы можно зафиксировать в отдельном представлении с помощью параметра **widget-instance-id**. Имя представления соответствует значению параметра.

При указании **widget-instance-id** в дополнительных параметрах, изменения, вносимые через другие IFrame-виджеты с той же страницей с графиками, не будут влиять на настройки, сохранённые в представлении.

В настройках IFrame-виджета можно менять значение параметра **widget-instance-id**, чтобы создать еще одно представление или восстановить настройки из ранее сохранённого.

Пример ссылки на страницу с графиками с использованием **widget-instance-id**:

http://127.0.0.1/iframe-widgets.html?#widget/selectable-chart/5fcc9a4de0a5393f45fb99f3/widget-instance-id=default

В данном примере:

- сервер работает на локальной машине и доступен по адресу 127.0.0.1;
- основным объектом указан объект с идентификатором **5fcc9a4de0a5393f45fb99f3**;

- настройки отображения страницы сохраняются в представлении с именем **default**.

3.11.3. Страница с журналом сессий

Журнал пользовательских сессий можно открывать по прямой ссылке, которая также может быть использована для встраивания в виджет IFrame:

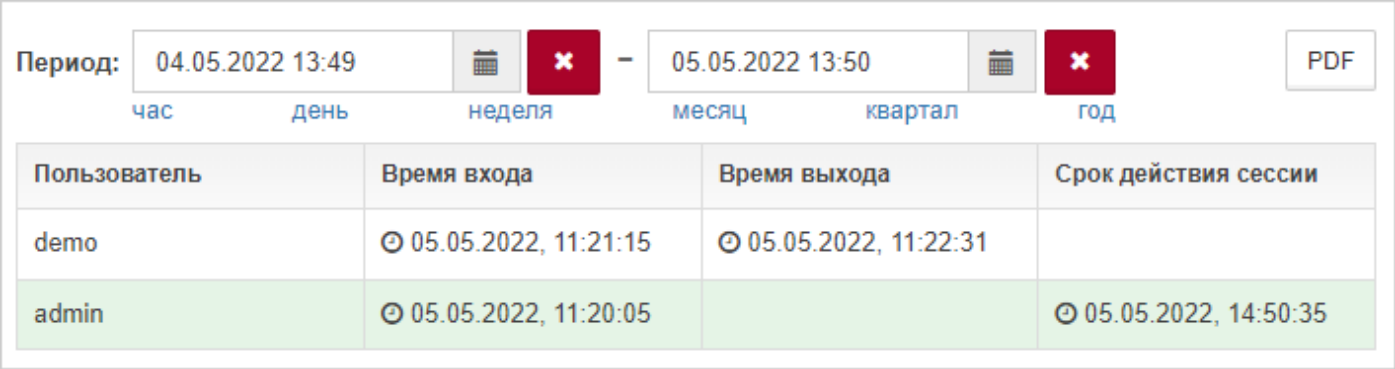


Рис. 47. Пример настроенной страницы с журналом сессий

Ссылка имеет следующую структуру:

*https://**<адрес сервера Центрального Пульта>**/iframe-widgets.html?#widget/session-log/1/**<дополнительные параметры>***

где:

- **<адрес сервера Центрального Пульта>** - IP-адрес или FQDN инсталляции Центрального Пульта;
- **<дополнительные параметры>** - настройки, которые определяют вид страницы.

Пример:

*https://**127.0.0.1**/iframe-widgets.html?#widget/session-log/1/**period=last-30days**&**limit=10***

В данном примере:

- сервер работает на локальной машине и доступен по адресу **127.0.0.1**;
- при загрузке страницы устанавливается временной период за последний месяц;
- выдача ограничена **10** записями.

Дополнительные параметры страницы журнала сессий:

Параметр	Описание
limit	Максимальное количество возвращаемых записей. По умолчанию выдача ограничена 30 записями.

Параметр	Описание
period	Период времени для отображения журнала сессий при загрузке страницы. По умолчанию отображаются сессии за последний час. Параметр может принимать следующие значения: • **period=last-hour** - за последний час, • **period=last-24hours** - за последний день, • **period=last-7days** - за последнюю неделю, • **period=last-30days** - за последний месяц, • **period=last-90days** - за последний квартал, • **period=last-365days** - за последний год.

3.11.4. Страница журнала событий

Страницу журнала событий можно открывать по прямой ссылке, которая также может быть использована для встраивания в виджет IFrame.

3.11.4.1. Параметры адресной строки

Фильтры и элементы интерфейса журнала событий можно настраивать с помощью параметров адресной строки.

Пример запроса с параметрами:

```
http://<saymon-server>/event-
log.html?type=mqtt&header=off&logo=on&pauseButton=on&typeSelect=on&filterSectio
n=on&exportSection=on
```

Список параметров адресной строки:

Параметр	Описание
type	Тип журнала, который будет отображаться по умолчанию. Можно изменить после загрузки страницы, если включены кнопки выбора типа журнала (`typeSelect`). Возможные значения: • `snmp-trap` – открывать режим SNMP трапы; • `mqtt` – открывать режим MQTT-сообщения; • `state-history` – открывать режим История состояний.
filter	Фильтры журнала. Зависят от выбранного типа журнала. Подробнее можно прочитать в разделе Фильтры.
header	Настраивает отображение шапки страницы, содержащей элементы управления фильтрацией данных в таблицах. • `on` – включить шапку; • `off` – выключить шапку. Также, автоматически присваивает значение `off` всем параметрам отображения элементов интерфейса. Чтобы перезаписать настройку отображения определённого элемента при выключенной шапке – например, выключить всё кроме логотипа – нужно выставить её в значение `on`: `http://<saymon-server>/event-log.html?header=off&logo=on` **TIP** Порядок параметров не имеет значения.

Параметр	Описание
logo	Настраивает отображение логотипа. • `on` – включить логотип, • `off` – выключить логотип.
typeSelect	Настраивает отображение кнопки выбора типа журнала (SNMP, MQTT или История состояний). • `on` – включить кнопку, • `off` – выключить кнопку.
filterSection	Настраивает отображение секции расширенного фильтра. • `on` – включить кнопку, • `off` – выключить кнопку.
paramsSection	Настраивает отображение секции дополнительных параметров (Количество сообщений для журнала SNMP трапов). • `on` – включить секцию, • `off` – выключить секцию.
exportSection	Настраивает отображение кнопки экспортирования данных в формате `.xls`. • `on` – включить кнопку, • `off` – выключить кнопку.
pauseButton	Настраивает отображение кнопки "Приостановить вывод" () • `on` – включить кнопку, • `off` – выключить кнопку.

Параметр	Описание
hideColumns	Позволяет скрыть столбцы из таблицы. Список возможных значений указан в разделе XREF_TEXT_NOT_FOUND. **WARNING** Этот параметр также применяется при экспорте таблицы. Скрытые столбцы не включаются в экспортированную таблицу.

Фильтры

Журнал Событий позволяет указывать параметры фильтра не только через меню "Фильтр", но и в адресной строке.

WARNING Фильтры, настроенные через пользовательский интерфейс, не применяются к адресной строке автоматически. При этом, фильтры заданные в адресной строке переносятся в меню "Фильтр", где их можно изменить.

```
http://<saymon-server>/event-log.html?filter=<тело фильтра>
```

TIP Запрос с использованием фильтра может содержать и другие параметры адресной строки.

Фильтры журналов MQTT сообщений и SNMP трапов поддерживают поиск по нескольким полям одновременно. Для этого, в параметр `filter` нужно передать список фильтров:

Запрос, который возвращает журнал MQTT сообщений с фильтром по топику и телу MQTT сообщения

```
http://<saymon-server>/event-
log.html?type=mqtt&filter=[{"name":"payload.topic","value":{"_ct":"2"}},
{"name":"payload.message","value":{"_ct":"message text"}}]
```

В запрос журнала истории состояний тоже можно передать список фильтров, но все, кроме первого, будут проигронированы.

Фильтры журнала истории состояний

Пример запроса с использованием фильтра:

```
http://<saymon-server>/event-
log.html?filter={"stateId":[1,3],"from":1599469080000,"to":1694077090252}
```

Фильтр поддерживает следующие параметры:

Параметр	Описание
`from`	Время начала выборки (в формате времени Unix, в милисекундах).
`to`	Время конца выборки (в формате времени Unix, в милисекундах).
`stateId`	ID или список ID состояний.
`classId`	ID или список ID классов объектов.
`tagId`	ID или список ID тегов.
`entityId`	ID объекта или связь. **Скрыт в интерфейсе**.
`entityType`	ID типа объекта (1 – объект, 2 – связь). **Скрыт в интерфейсе**.
`entityName`	Фильтр по частичному совпадению имени. **Скрыт в интерфейсе**.

Фильтры журнала MQTT сообщений

Пример фильтра журнала MQTT сообщений:

```
http://<saymon-server>/event-
log.html?filter=[{"name":"payload.topic","value":{"_ct":"2"}}]
```

Параметр	Описание
`name`	Название переменной, по значению которой нужно фильтровать журнал. У MQTT сообщений есть следующие переменные: • `payload.topic` – топик MQTT сообщения; • `payload.message` – тело MQTT сообщения.
`value`	Фильтр значения указанной переменной в следующем формате: `"value":{<оператор>:<значение>}` Поддерживаемые операторы: • `_ct` – Содержит; • `_nct` – Не содержит; • `_eq` – Равен; • `_neq` – Не равно; • `_m` – Совпадает с.

Фильтры журнала SNMP трапов

Пример фильтра журнала SNMP трапов:

```
http://<saymon-server>/event-
log.html?filter=[{"name":"payload.text","value":{"_ct":"34"}}]
```

Параметр	Описание
`name`	Название переменной, по значению которой нужно фильтровать журнал. У MQTT сообщений есть следующие переменные: • `payload.trapOid` • `payload.text` • `payload.senderAddress`

Параметр	Описание
`value`	Фильтр значения указанной переменной в следующем формате: `"value":{<оператор>:<значение>}` Поддерживаемые операторы: • `_ct` – Содержит; • `_nct` – Не содержит; • `_eq` – Равен; • `_neq` – Не равно; • `_m` – Совпадает с.

3.12. Открытие терминала удалённого доступа

Для объектов класса "Host", а также для объектов с хотя бы одним из заданных свойств

- address (приоритетнее);
- IP,

в контекстном меню доступны дополнительные пункты:

- "Открыть SSH терминал",
- "Открыть telnet терминал".

Если IP-адрес указан в свойствах объекта, при выборе соответствующего пункта меню происходит подключение к терминалу.

По умолчанию используются следующие порты:

- 22 для SSH;
- 23 для telnet.

При необходимости использовать другой порт номер порта указывается в свойствах объекта для соответствующего протокола:

- ssh_port
- telnet_port

Логин пользователя также можно заранее указать в свойствах для соответствующего протокола:

- ssh_login
- telnet_login

Для объектов класса "Host", если в свойствах объекта IP-адрес не задан, при выборе соответствующего пункта меню открывается окно (Пример окна подключения к SSH-терминалу), в котором необходимо:

- ввести IP-адрес объекта в поле "Адрес";
- указать номер порта в поле "Порт";
- ввести логин пользователя в поле "Пользователь".

WARNING	*При включенном слайдере "Сохранить" введённые данные сохранятся в свойствах объекта.*

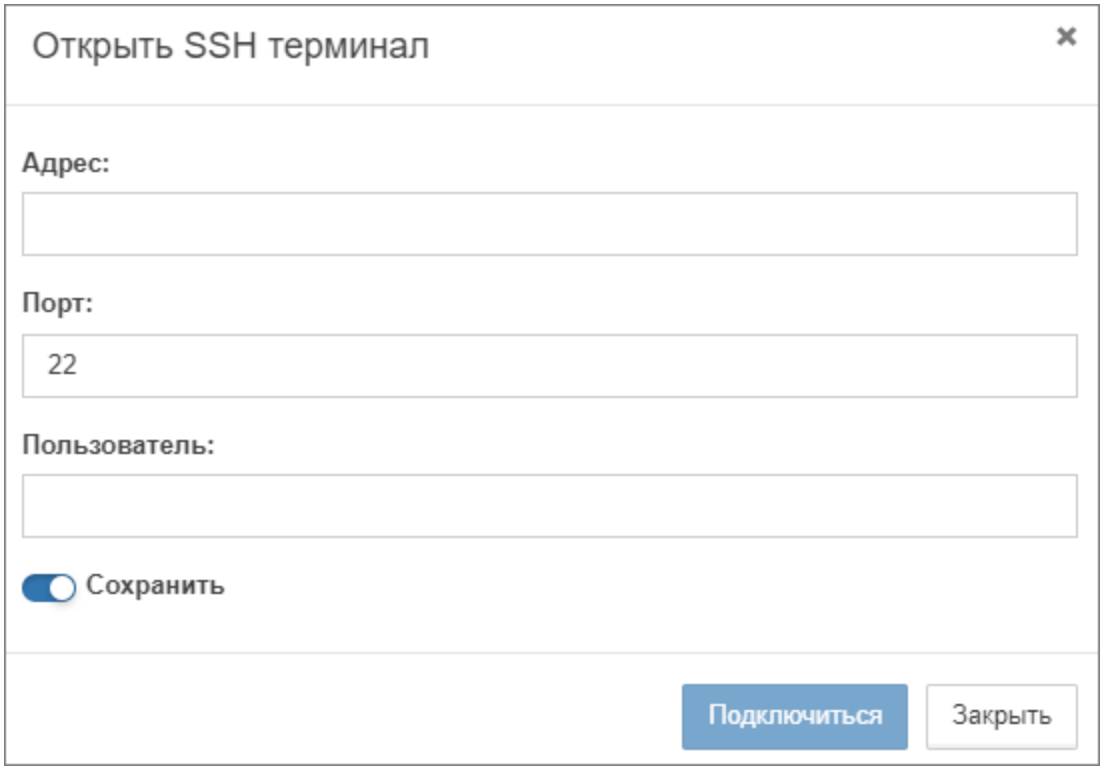


Рис. 48. Пример окна подключения к SSH-терминалу

После ввода данных необходимо нажать кнопку Подключиться.

3.13. Переход к web-интерфейсу устройства

Для объектов с хотя бы одним из следующих свойств (в порядке уменьшения приоритета)

- web_link,
- address,
- IP,

в контекстном меню доступен пункт "Переход по ссылке". При выборе данного пункта

происходит переход по указанному в значении свойства IP-адресу или URL.

3.14. Резервное копирование и восстановление данных

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

3.14.1. MongoDB

MongoDB - NoSQL-СУБД, которая является основным хранилищем данных о пользователях, объектах и связях, их свойствах, состояниях и авариях.

3.14.1.1. Резервирование

Для экспорта данных из MongoDB в архив необходимо создать и выполнить следующий скрипт:

```
$ sudo /opt/saymon-scripts/mongodump.sh
```

Пример текста скрипта (необходимо указать `HOST`, `LOGIN`, `PASS`):

```
#!/bin/bash
BACKUP=/backups/mongo/mongodump-$(date +%F-%H-%M).gz
HOST='10.130.0.10'
LOGIN='saymon'
PASS='5tr43ewQ21'

# Outputs a log message, prefixed with current date-time.
log() {
    echo "[`date`] $1"
}

# Create MongoDB backup.
mongodump -h"$HOST" -u"$LOGIN" -p"$PASS" --db saymon --gzip --archive >
"$BACKUP"

log "MongoDB stored to $BACKUP"
```

По умолчанию резервная копия создается в директории `/backups/mongo` в формате **gz**.

Для периодического автоматического создания новых и удаления старых резервных копий можно добавить соответствующие команды в crontab, например:

```
$ sudo crontab -e

0 */2 * * * /opt/saymon-scripts/mongodump.sh
```

```
10 */2 * * * ls -tr /backups/mongo | xargs realpath | head -n -6 | xargs rm
```

3.14.1.2. Восстановление

Для восстановления или импорта данных в MongoDB необходимо создать и выполнить следующий скрипт:

```
$ sudo /opt/saymon-scripts/mongorestore.sh
```

Пример текста скрипта (необходимо указать HOST, LOGIN, PASS):

```
#!/bin/bash
HOST='10.130.0.10'
LOGIN='saymon'
PASS='5tr43ewQ21'

# Outputs a log message, prefixed with current date-time.
log() {
    echo "[`date`] $1"
}

# Directory for MongoDB backups.
DIR=/backups/mongo

# Ask user to specify dump file.
read  -p "Please specify backup file, e.g. mongodump-2021-04-15-14-10.gz: "
BACKUP

# Check if BACKUP exists, remove DIR if ndeeded.
if [[ "$BACKUP" = */backups/mongo/* ]]; then
  BACKUP=$(sed 's/\/backups\/mongo\///' <<< "$BACKUP")
fi

if [ -f $DIR/$BACKUP ]; then
  log "Backup file $DIR/$BACKUP found!"
else
  log "Backup file $DIR/$BACKUP not found!"
  exit 1
fi

# Create MongoDB backup.
mongorestore -h"$HOST" -u"$LOGIN" -p"$PASS" --db saymon --drop --noIndexRestore
--gzip --archive="$DIR/$BACKUP"
redis-cli -h 10.130.0.10 -a '$6$9UG8HCr0nqj$bVMZfvDKTobl' flushall

log "MongoDB, database saymon, restored from $DIR/$BACKUP"
```

3.14.2. OpenTSDB

OpenTSDB — специализированное хранилище временных рядов, в котором хранятся числовые метрики, используемые при построении графиков.

3.14.2.1. Резервирование

Для экспорта данных из OpenTSDB в архив на существующей инсталляции системы необходимо создать и выполнить следующий скрипт:

```
$ sudo /opt/saymon-scripts/hbasedump.sh
```

WARNING

Директория с данными OpenTSDB может отличаться в разных инсталляциях сервера. Перед созданием скрипта убедитесь в наличии данных в директории, указанной в переменной `SOURCE`.

Данные OpenTSDB могут храниться в следующих директориях:

- `/var/lib/hbase`
- `/data`
- `/var/lib/docker/volumes/<volume_id>/_data/hbase-root/hbase`

Также, папку `hbase` можно попробовать найти командой:

```
$ sudo find / -type d -name 'hbase'
```

Пример текста скрипта:

```
#!/bin/bash
SOURCE=/var/lib/hbase
BACKUP=/backups/hbase/hbasedump-$(date +%F-%H-%M).tar.gz

# Outputs a log message, prefixed with current date-time.
log() {
    echo "[`date`] $1"
}

# Create HBase backup.
tar zcvf "$BACKUP" "$SOURCE"

log "HBase stored to $BACKUP"
```

По умолчанию резервная копия создается в директории `/backups/hbase` в формате **.tar.gz**.

NOTE	Директория, в которую сохраняются резервные копии должна существовать перед выполнением скрипта.

Для периодического автоматического создания новых и удаления старых резервных копий можно добавить соответствующие команды в crontab, например:

```
$ sudo crontab -e

0 */2 * * * /opt/saymon-scripts/hbasedump.sh
10 */2 * * * ls -tr /backups/hbase | xargs realpath | head -n -6 | xargs rm
```

3.14.2.2. Восстановление

Для восстановления или импорта данных в OpenTSDB на новой инсталляции системы необходимо создать и выполнить следующий скрипт:

```
$ sudo /opt/saymon-scripts/hbaserestore.sh
```

WARNING	Директория с данными OpenTSDB может отличаться в разных инсталляциях сервера. Перед созданием скрипта убедитесь в правильности директории, указанной в переменной DEST.

Текст скрипта:

```
#!/bin/bash
#
# Outputs a log message, prefixed with current date-time.
log() {
    echo "[`date`] $1"
}

# Directory for OpenTSDB backups.
DIR=/backups/hbase

# Where to restore OpenTSDB data.
DEST=/var/lib/hbase

# Ask user to specify dump file.
read  -p "Please specify backup file, e.g. hbasedump-2021-04-15-14-31.tar.gz: "
BACKUP

# Check if BACKUP exists, remove DIR if ndeeded.
if [[ "$BACKUP" = */backups/hbase/* ]]; then
  BACKUP=$(sed 's/\/backups\/hbase\///' <<< "$BACKUP")
fi

if [ -f $DIR/$BACKUP ]; then
```

```
  log "Backup file $DIR/$BACKUP found!"
else
  log "Backup file $DIR/$BACKUP not found!"
  exit 1
fi

# Restore from BACKUP.
docker stop opentsdb
rm -rf "$DEST"
mkdir -p "$DEST" && tar xvf $DIR/$BACKUP -C /
docker start opentsdb

log "HBase restored from $DIR/$BACKUP"
```

3.14.3. Конфигурационные файлы

Для экспорта настроек конфигурации необходимо сделать копию директории с конфигурационными файлами `/etc/saymon`.

Для импорта настроек конфигурации необходимо поместить файлы из ранее сохранённой директории с конфигурационными файлами в директорию `/etc/saymon`.

3.15. Расширения

Центральный Пульт поддерживает установку расширений, дополняющих функционал системы.

3.15.1. Установка расширений

К списку установленных расширений имеют доступ все пользователи через меню текущего пользователя (**Имя пользователя › Расширения**), но доступ к данным ограничен их правами и доступом к объектам.

Для добавления нового расширения необходимо нажать кнопку 

или , или перейти на страницу saymon.tech/extensions.

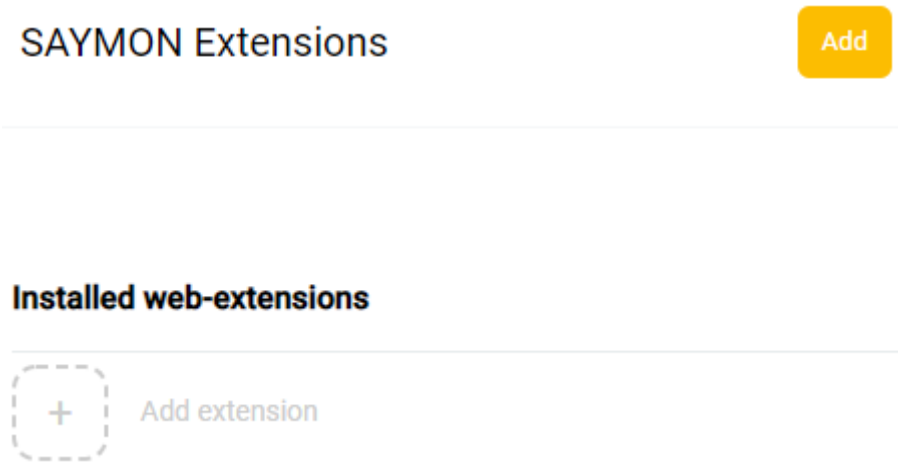


Рис. 49. Меню расширений

В открывшемся каталоге перейти на страницу интересующего расширения и выполнить шаги, описанные в инструкции по установке.

3.15.2. Пользовательские расширения

Пользователи могут писать собственные web-приложения, расширяющие или использующее возможности Центрального Пульта.

Чтобы включить такие приложения в состав сервера Центрального Пульта, необходимо:

1. Добавить секцию **extensions** в конфигурационный файл сервера и в параметре `path` указать путь к папке с расширениями, например, `"/opt/saymon-extensions"`:

   ```
   "extensions": {
       "path": "/opt/saymon-extensions"
   }
   ```

2. Расположить содержимое каждого расширения в отдельной папке внутри корневой. Например, `/opt/saymon-extensions/my-extension`. Эта папка будет точкой входа в клиентскую часть расширения, доступную по URL: `<Адрес_Сервера_Центрального_Пульта>/extensions/my-extension`.
3. В файле `/etc/nginx/sites-enabled/saymon` добавить секцию `location ^~ /extensions` и указать в параметре `alias` папку из `extensions.path`:

   ```
   # SAYMON Extensions folder.
       location ^~ /extensions {
          alias /opt/saymon-extensions/;
          try_files $uri $uri$args $uri$args/ @extension;
          autoindex on;
       }
   ```

Чтобы расширение было доступно в общем списке расширений, отображаемых в web-интерфейсе Центрального Пульта, оно должно содержать файл `package.json` или

`meta.json` (данные `meta.json` приоритетнее).

Формат и поля файла `meta.json`:

Параметр	Описание
author	Автор расширения.
date	Дата выхода установленного расширения.
description	Краткое описание расширения.
homepage	Ссылка на домашнюю страницу расширения.
icon	Прямая ссылка на иконку расширения. По умолчанию используется файл `icon.png` из директории расширения.
name	Название расширения. **WARNING** Расширения сторонних разработчиков не могут начинаться со слова **SAYMON**.
nameInUrl	Часть ссылки для перехода на страницу с расширением. Совпадает с названием папки, в которой лежит файл `index.html` расширения. `<Адрес_Сервера_Центрального_Пульта>/extensions/<nameInUrl>`
urlToMeta	Прямая ссылка на файл `meta.json` последней актуальной версии для проверки обновлений.
version	Версия установленного расширения.

Пример файла `meta.json`:

```
{
    "author": "SAYMON User",
    "date": "2020/12/31",
    "description": "Very useful custom extension",
    "homepage": "https://saymon.tech/extensions/my-extension/",
    "icon": "https://saas.saymon.info/extensions/my-extension/favicon.ico",
    "name": "My Extension",
    "nameInUrl": "my-extension",
    "urlToMeta": "https://saymon.tech/extensions_meta/my-extension/meta.json",
    "version": "1.0.1"
}
```

4. 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.

4.1. 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 **
   - */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 описанных выше 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
```

4.2. 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
```

4.3. 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 stop
sudo service mongod start
sudo service mongod restart
```

4.4. 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
sudo service redis-server start
sudo service redis-server 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"
```

4.5. 500 Internal Server Error и отсутствие графиков

Условия проблемы:

- вместо графиков возникает ошибка 500.

Решение проблемы:

Необходимо перезапустить OpenTSDB:

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

4.6. Ошибка работы HTTP-проверки

Условия 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.

Есть два 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.

Приложение А (обязательное)

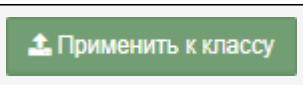
Права на операции

Основные права:

Название	**Описание**	**Запись в конфигурационном файле сервера**
Выполнение операций	Включает секцию **Операции** и возможность их запуска.	"execute-operations"
Поиск и групповые операции	Включает возможность выполнения поиска и групповых операций. *Необходимы права на "Управление объектами, свойствами и документами"*	"run-bulks

Название		Описание	Запись в конфигурационном файле сервера
Управление объектами, свойствами и документами		Включает • создание, • клонирование, • переименование, • изменение размеров, • перемещение, • удаление • объектов, • виджетов объектов, • ссылок, • подложек. Включает просмотр и редактирование всех секций объектов, кроме секции **Мониторинг**.	"manage-objects"
	Создание объектов	Включает • создание, • клонирование объектов.	"create-objects"
	Модификация объектов	Включает • переименование, • изменение размеров, • перемещение • объектов, • виджетов объектов, • ссылок, • подложек.	"modify-objects"
	Удаление объектов	Включает удаление объектов.	"delete-objects"

Название		Описание	Запись в конфигурационном файле сервера
Управление связями, свойствами и документами		Включает • создание, • изменение, • удаление • связей, • виджетов связей. Включает просмотр и редактирование всех секций связей, кроме секции **Мониторинг**.	"manage-links"
	Создание связей	Включает создание связей.	"create-links"
	Модификация связей	Включает • изменение связей, • создание, изменение и удаление виджетов связей.	"modify-links"
	Удаление связей	Включает удаление связей.	"delete-links"
Управление потоками		Включает создание, изменение и удаление потоков. *Для изменения потоков требуются права на просмотр секции "Параметры".*	"manage-flows"
	Создание потоков	Включает создание потоков.	"create-flows"
	Модификация потоков	Включает изменение потоков. *Требуются права на просмотр секции "Параметры".*	"modify-flows"
	Удаление потоков	Включает удаление потоков.	"delete-flows"

Название		Описание	Запись в конфигурационном файле сервера
Управление документами объектов и связей		Включает редактирование секции "Документы". *Требуются права на просмотр секции "Документы".*	"manage-documents"
	Создание документов	Включает прикрепление ссылок на web-страницы.	"create-documents"
	Модификация документов	Включает редактирование документов.	"modify-documents"
	Удаление документов	Включает удаление документов.	"delete-documents"
	Загрузка документов	Включает прикрепление PDF-файлов.	"upload-documents"
Управление свойствами объектов и связей		Включает редактирование секции **Свойства**. *Требуются права на просмотр секции "Свойства".*	"manage-properties"
	Создание свойств	Включает создание свойств.	"create-properties"
	Модификация свойств	Включает модификацию свойств.	"modify-properties"
	Удаление свойств	Включает удаление свойств.	"delete-properties"
Управление классами		Включает кнопку  в секциях • Условия перехода состояний, • Условия генерации аварий. Включает в окне конфигурации системы вкладку **Управление классами**.	"manage-classes"

Название	Описание	Запись в конфигурационном файле сервера
Управление операциями	Включает редактирование секции **Операции**.	"manage-operations"
Управление настройками мониторинга	Включает просмотр и редактирование секции **Мониторинг**.	"manage-service-properties"
Управление скриптами	Включает возможность использовать скрипт с указанным текстом при проверке **Выполнение программы / скрипта**. Включает в окне конфигурации системы вкладку **Скрипты**.	"manage-scripts"

Просмотр секций:

Секция	Запись в конфигурационном файле сервера
Данные	"view-section-stat"
Свойства	"view-section-properties"
Документы	"view-section-documents"
История состояний	"view-section-state-history"
Графики	"view-section-history-graph"
Мониторинг	"view-section-monitoring"
Параметры	"view-section-entity-settings"
Условия перехода состояний	"view-section-state-conditions"
Действия при смене состояний	"view-section-state-triggers"
Виджеты	"view-section-widgets"
История изменений	"view-section-audit-log"
Изменения конфигурации	"view-section-config-log"
Операции	"view-section-operations"

Секция	Запись в конфигурационном файле сервера
История операций	"view-section-operations-history
Условия генерации аварий	"view-section-incident-conditions"
Правила формирования данных	"view-section-stat-rules"

Права на администрирование системы:

Название	Описание	Запись в конфигурационном файле сервера
Управление конфигурацией системы	Включает в окне конфигурации системы вкладки: • Журнал событий, • Шаблоны уведомлений, • Аварии, • Состояния, • Парольные политики.	"manage-configuration"
Обновление агентов	Позволяет пользователю обновлять отдельных агентов кнопкой и всех агентов на вкладке **Агенты** в окне конфигурации системы.	"upload-agent-updates"
Управление пользователями	Позволяет управлять всеми пользователями системы.	"manage-users"
Управление журналом событий	Позволяет • назначать ответственного, • изменять текст событий, • удалять события в журнале событий SNMP.	"manage-event-log"

Название	Описание	Запись в конфигурационном файле сервера
Управление историческими данными	Позволяет • добавлять, • редактировать, • удалять комментарии на графиках.	"manage-history"
Запуск автоопределения объектов	Позволяет использовать сервисный REST-метод **node/api/discovery**.	"run-discovery"
Чтение журнала сессий	Включает в окне конфигурации системы вкладку **Журнал сессий**.	"read-session-log"

Приложение Б (обязательное)

Параметры исполняемых файлов пользовательских сенсоров и скриптов, выполняемых при создании/удалении объектов

Поле	Описание
title	Отображаемое в web-интерфейсе имя исполняемого файла. Опциональное поле. По умолчанию - идентификатор сенсора или скрипта. *Если идентификатор содержит путь (исполняемый файл вложен в подпапки), имя будет отображаться с путём.* Пример: *путь к файлу: (путь до папки с исполняемыми файлами)/cisco/task_1.sh отображаемое имя скрипта: cisco - task_1*
excludeForClasses	Список ID классов объектов, к которым неприменим данный сенсор. Опциональное поле. По умолчанию сенсор доступен для всех классов объектов. ***Поле применяется только для пользовательских мониторинговых сенсоров.***

Поле	Описание
icon	Иконка из набора Font Awesome, отображаемая рядом с именем проверки. Опциональное поле. По умолчанию -  - 'fa fa-cubes fa-fw' ***Поле применяется только для пользовательских мониторинговых сенсоров.***
args	Настройки аргументов, которые задаются пользователем через web-интерфейс и передаются исполняемому файлу.
args.id	Имя аргумента, передаваемое исполняемому файлу. Опциональное поле.
args.default	Значение аргумента по умолчанию, подставляемое в соответствующее поле. Опциональное поле.
args.description	Описание аргумента, отображаемое в web-интерфейсе. Опциональное поле.
args.name	Название аргумента, отображаемое в web-интерфейсе.
args.required	Признак обязательности заполнения значения аргумента. • false - значение опционально, • true - значение обязательно. Опциональное поле. По умолчанию – **false.**

Поле	Описание
args.options	Применимо только для типа **select** (*args.type.select*). Определяет набор значений, отображаемых в выпадающем списке. По умолчанию при создании проверки выбирается первая опция. Чтобы установить иную опцию значением по умолчанию, необходимо указать значение поля *args.options.value* в поле *args.default*. Исполняемому файлу передаются: • имя аргумента (*args.id*); • значение аргумента (*args.options.value*). Если *args.id* не задано, то исполняемому файлу передаётся только значение выбранного аргумента (*args.options.value*). Если значение аргумента не задано (*args.options.value*), то исполняемому файлу также не передаётся имя аргумента (*args.id*). *Данную логику можно использовать для реализации пункта "Не выбрано" и разделителей в выпадающем списке.*
args.options. description	Описание значения аргумента, отображаемое в выпадающем списке. Опциональное поле. По умолчанию отображается значение аргумента (*args.options.value*).
args.options.value	Значение аргумента. Опциональное поле. При отсутствии значения исполняемому файлу также не передаётся имя аргумента. *Такие опции можно использовать для реализации пункта "Не выбрано" и разделителей в выпадающем списке.*
args.type	Тип значения аргумента. Опциональное поле. По умолчанию - text.

Поле	Описание
args.type.checkbox	Переключатель. По умолчанию выключен. При включении исполняемому файлу передаётся значение, указанное в поле *args.id* (имя аргумента). Если это поле отсутствует, то исполняемому файлу передаётся порядковый номер аргумента (отсчёт начинается с 0).
args.type.password	Отображает текстовое поле для ввода значения аргумента с маскировкой введённых символов.
args.type.select	Отображает выпадающий список с предопределёнными значениями аргумента. Список значений задается в поле *args.options*. Для данного типа не отображается описание аргумента (поле *args.description*). Вместо этого необходимо использовать соответствующее поле для каждого значения аргумента (*args.options.description*).
args.type.text	Отображает текстовое поле для ввода значения аргумента.
args.type.textarea	Отображает текстовое поле для ввода значения аргумента. Позволяет вводить многострочный текст.

Пример скрипта

Для демонстрации работы функционала скриптов и различных типов аргументов мы подготовили два файла.

1. Разместите их в каталоге скриптов.
2. Настройте выполнение скрипта "Custom script example" при создании или удалении объекта любого класса во вкладке Триггеры.
3. Задайте значения аргументов.
4. Создайте или удалите объект выбранного класса.

Исполняемый файл скрипта записывает все переданные в него аргументы в файл *output*:

hello_world.sh

```
#!/bin/bash
```

```
#
# Use this script as the custom one
# to check or test arguments configuration file.

i=0
echo "Script got the following args:"
for arg in $*
do
  i=$((i + 1))
  echo -ne $arg >> /opt/saymon-extensions/entity-triggers-path/myselect/output;
echo -ne " " >> /opt/saymon-extensions/entity-triggers-path/myselect/output
done

if [ $i = 0 ]
then
  echo "(no args)" >> /opt/saymon-extensions/entity-triggers-
path/myselect/output
  exit 2
fi
```

В этом конфигурационном файле приведены примеры для всех типов аргументов:

hello_world.sh.conf

```
{
    "title": "Custom script example",
    "args": [
        {
            "name": "Unnamed (positional) arg"
        },
        {
            "name": "Named arg",
            "id": "--name"
        },
        {
            "name": "Behold the description on the right",
            "description": "Here I am!",
            "id": "--description"
        },
        {
            "name": "Arg with default value",
            "description": "and description",
            "id": "--defval",
            "default": "default_value"
        },
        {
            "name": "Required arg",
            "description": "This field is required to fill in",
            "id": "--required",
            "required": true
        },
```

```
        {
            "name": "Type - Text",
            "description": "This and all above fields have got text type",
            "id": "--text",
            "type": "text"
        },
        {
            "name": "Type - Text Area",
            "description": "This field is text area. \nIt is resizable in most
browsers. \nAnd allows to enter multiline texts.",
            "id": "--textarea",
            "type": "textarea"
        },
        {
            "name": "Type - Checkbox",
            "description": "This is the checkbox type",
            "id": "--checkbox",
            "type": "checkbox"
        },
        {
            "name": "Type - Password",
            "description": "This is the password field",
            "id": "--pass",
            "default": "qwerty",
            "type": "password"
        },
        {
            "name": "Type - Select",
            "description": "This is the select type. This description is not
displayed in the web interface. Use description for each option instead.",
            "id": "--select",
            "default": "option2",
            "type": "select",
            "options": [
                {
                    "description": "== Not selected =="
                },
                {
                    "value": "option1", "description": "OK"
                },
                {
                    "description": "== Divider =="
                },
                {
                    "value": "option2", "description": "Warning"
                },
                {
                    "value": "option3", "description": "Error"
                }
            ]
        }
    ]
```

```
}
```

Приложение В (обязательное)

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.

agent_stat

В этом разделе задаются настройки хранения истории данных от агента.

```
"agent_stat": {
   "write_to": "mongo",
   "default_ttl": 20,
   "ttl": [
      {
        "key": "payload.cpuInformation.vendor",
        "value": "Intel",
        "expire": 10
      }
   ]
}
```

agent_stat.**write_to**

Выбор хранилища данных от агента.

При указании значения "mongo" данные от агента хранятся в MongoDB.

По умолчанию – данные от агента хранятся в ELK.

agent_stat.**default_ttl**

Срок жизни наборов данных (в секундах) по умолчанию.

Если это значение равно `0`, то данные не записываются в БД.

По умолчанию – `86400`

agent_stat.**ttl**

Сроки жизни наборов данных, для которых нужен срок жизни, отличающийся от принятого по умолчанию.

agent_stat.ttl.**key**

Имя метрики.

agent_stat.ttl.**value**

Значение метрики.

agent_stat.ttl.**expire**

Срок жизни набора данных.

Если это значение равно `0`, то данные не записываются в БД.

cache

В этом разделе задаются настройки сервера Redis.

```
"cache": {
    "host": "127.0.0.1",
    "port": 6379
}
```

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`.

cache.**cluster**

В этом подразделе задаются настройки при использовании Redis Cluster.

```
"cache": {
    "cluster": {
        "nodes": [
            {
                "port": 7000,
                "host": "127.0.0.1"
            },
            {
                "port": 7001,
                "host": "127.0.0.1"
            }
        ],
        "options": {}
    }
}
```

WARNING	Если задействован этот подраздел, система использует указанные здесь настройки подключения, параметры **cache.host** и **cache.port** игнорируются.

cache.cluster.**options**

Дополнительные параметры подключения к кластеру.

Подробнее можно узнать в документации ioredis.

cache.cluster.**nodes**

Серверы кластера, к которым необходимо подключиться.

cache.cluster.nodes.**host**

Адрес сервера кластера.

cache.cluster.nodes.**port**

Порт сервера кластера.

custom_scripts

В этом разделе задаются настройки пользовательских скриптов, выполняющихся при создании и удалении объектов.

```
"custom_scripts": {
    "entity_triggers_path": "/opt/saymon-extensions/entity-triggers-path"
}
```

custom_scripts.**entity_triggers_path**

Полный путь до папки с файлами скриптов.

db

В этом разделе задаются настройки подключения к базе данных для хранения объектной модели. В качестве базы данных используется MongoDB.

```
"db" : {
    "mongodb": {
        "url": "mongodb://localhost:27017/saymon?w=1"
    }
}
```

db.**mongodb**

Данные для подключения к MongoDB.

db.mongodb.**url**

Адрес сервера MongoDB и дополнительные параметры подключения.

По умолчанию – `"mongodb://localhost:27017/saymon?w=1"`.

WARNING	При использовании кластера из нескольких хостов их адреса указываются через запятую в произвольном порядке: `"url": "mongodb://host01:27017,host02:27017,host03:27017/saymon?w=1"`

Все возможные дополнительные параметры URL представлены в документации по MongoDB.

elasticsearch

В этом разделе задаются настройки для работы с Elasticsearch.

```
"elasticsearch": {
    "node": "http://127.0.0.1:9200",
    "auth": {
        "username": "elastic",
        "password": "ypezA1ZoVv46fJh8Abj4"
   },
   "index_pattern": "myindex*",
   "request_timeout": 30000,
   "hits_per_request": 1000
```

```
}
```

elasticsearch.**hits_per_request**

Максимальное число записей, запрашиваемых в одном обращении к elasticsearch.

WARNING	При повышении значения параметра снижается число запросов к Elasticsearch, но увеличивается расход памяти и время блокировки потока на сервере Центрального Пульта при обработке ответа.

По умолчанию – `1000`.

elasticsearch.**index_pattern**

Индекс Elasticsearch, в котором происходит поиск.

elasticsearch.**node**

IP-адрес инстанса Elasticsearch.

elasticsearch.**request_timeout**

Время ожидания ответа от сервера elasticsearch (в миллисекундах).

По умолчанию – `30000` (30 секунд).

elasticsearch.**auth**

Авторизационные данные пользователя elasticsearch.

elasticsearch.auth.**password**

Пароль пользователя elasticsearch.

elasticsearch.auth.**username**

Логин пользователя elasticsearch.

encrypt

```
"encrypt": {
  "db": true,
  "server_config": true,
  "tags": [
    "<<",
    ">>"
  ],
  "prefix": "crypt"
```

```
}
```

encrypt.**db**

Включает шифрование полей в базе данных.

- true – Включить шифрование.
- false – Выключить шифрование.

По умолчанию – `false`

encrypt.**server_config**

Включает шифрование полей в конфигурационном файле сервера

- true – Включить шифрование.
- false – Выключить шифрование.

По умолчанию – `false`

encrypt.**tags**

Тег для обозначения зашифрованного поля или поля, которое необходимо зашифровать. Подробнее о формате зашифрованного поля можно узнать в разделе Формат зашифрованных параметров.

По умолчанию – `["<<",">>"]`

encrypt.**prefix**

Префикс, обозначающий, что поле зашифровано. Подробнее о формате зашифрованного поля можно узнать в разделе Формат зашифрованных параметров.

По умолчанию – `crypt`

extensions

В этом разделе задаются параметры пользовательских расширений Центрального Пульта.

```
"extensions": {
    "path": "/opt/saymon-extensions"
}
```

extensions.**path**

Путь к папке с пользовательскими расширениями.

hostname_update

В этом разделе задаются настройки автоматического обновления информации автоматически зарегистрированных объектов класса `Host`.

```
hostname_update: {
    enabled: false
}
```

hostname_update.**enabled**

Автоматическое обновление имени и свойств на объекте класса `Host` при смене имени хоста.

- true – Включить автообновление.
- false – Выключить автообновление.

По умолчанию – `false`

kafka

В этом разделе задаются настройки подключения к брокеру Kafka.

```
"kafka" : {
    "host" : "localhost",
    "port" : 9092,
    "requestTimeout" : 30000,
    "fetch_latest" : false,
    "logLevel" : 2
}
```

Помимо перечисленных здесь параметров, возможно также использовать дополнительные параметры подключения.

kafka.**brokers**

Пул адресов для подключения при использовании Kafka Cluster.

```
"kafka" : {
   ...
   "brokers": [
      "192.168.1.10:9092",
      "192.168.1.11:9092"
   ],
   ...
}
```

WARNING	При указании данного параметра система игнорирует параметры **kafka.host** и **kafka.port**.

WARNING	При использовании Kafka Cluster необходимо дополнительно настроить конфигурационный файл агента.

kafka.**fetch_latest**

Данный параметр определяет правило обработки сообщений, поступивших в Kafka при выключенной системе:

- false – обработать накопленные данные,
- true – игнорировать накопленные данные.

По умолчанию – `false`.

kafka.**host**

Адрес брокера Kafka.

По умолчанию – `"localhost"`.

kafka.**logLevel**

Уровень логирования Kafka;

- 0 – NOTHING,
- 1 – ERROR,
- 2 – WARN,
- 4 – INFO,
- 5 – DEBUG.

По умолчанию – 2 – WARN.

kafka.**port**

Порт брокера Kafka.

По умолчанию – `9092`.

kafka.**requestTimeout**

Время на ожидание ответа от брокера (в миллисекундах).

По умолчанию – `30000` – 30 секунд.

kafka.ssl

Настройки шифрования.

Ключ и сертификат можно указывать как путь к файлу:

```
"kafka" : {
...
    "ssl": {
        "certFile": "/opt/kafka/for-saymon-server/cert.pem",
        "keyFile": "/opt/kafka/for-saymon-server/key.pem"
    },
...
}
```

kafka.sasl

Настройки аутентификации. Параметры зависят от механизма аутентификации, используемом в Kafka.

Пример настроек для механизма PLAIN/SCRAM:

```
"kafka" : {
...
    "sasl": {
        "mechanism": "plain",
        "username": "user",
        "password": "password"
    },
...
}
```

kafka.sasl.mechanism

Механизм аутентификации.

keycloak

В этом разделе задаются параметры для взаимодействия с сервером Keycloak для авторизации пользователей.

```
"keycloak": {
    "realm_certificate": "<data of keycloak_realm_certificate>",
    "realm_certificate_file": "/etc/saymon/keycloak_realm_certificate",
    "config_file": "/etc/saymon/keycloak_backend.json"
}
```

WARNING	Параметры в примере перечислены в порядке понижения приоритета при одновременном указании. Для работы с Keycloak достаточно указать один из параметров.

keycloak.**config_file**

Путь к файлу конфигурации, сгенерированному сервером Keycloak для Backend-клиента.

keycloak.**realm_certificate**

Данные сертификата (Realm Certificate), которым сервер Keycloak подписывает пользовательские токены.

keycloak.**realm_certificate_file**

Путь к файлу с Keycloak Realm Certificate.

ldap

В этом разделе задаются параметры внешнего LDAP-сервера для авторизации пользователей.

```
"ldap": {
    "url": "ldaps://192.168.1.1:636",
    "suffix": "dc=example,dc=com",
    "login": "cn=admin,dc=example,dc=com",
    "pass": "root",
    "allow_self_signed": true,
    "create_user_for_existing_group_only": true,
    "import_non_existing_groups": false,
    "update_existing_groups": false
}
```

ldap.**allow_self_signed**

Разрешает соединение по LDAPs с сервером, на котором установлен самоподписанный SSL-сертификат;

- false – соединение запрещено,
- true – соединение разрешено.

По умолчанию – `false` – соединение запрещено.

ldap.**create_user_for_existing_group_only**

Ограничивает авторизацию новых LDAP-пользователей только теми пользователями, у которых имя группы на LDAP-сервере совпадает с именем существующей (созданной

ранее) группы в Центральном Пульте;

- false – разрешена авторизация всех новых LDAP-пользователей,
- true – разрешена авторизация только тех новых LDAP-пользователей, для которых заготовлены группы в Центральном пульте.

По умолчанию – `false` – разрешена авторизация всех новых LDAP-пользователей.

WARNING	На LDAP-пользователей, у которых уже есть учётная запись, применение данного параметра не распространяется. Нельзя использовать вместе с параметром `import_non_existing_groups`. Если у обоих параметров значение `true`, то сервер сначала импортирует группу нового пользователя, а потом авторизует его, так как он входит в существующую группу.

ldap.**group_name_attribute**

Атрибут, в котором хранятся имена групп пользователей.

Указывается, если необходимо использовать нестандартный атрибут.

Стандартный атрибут – **"cn"**.

ldap.**import_non_existing_groups**

Разрешает автоматическое создание групп, в которых пользователь состоит на LDAP-сервере, но которых нет в Центральном Пульте;

- false – при авторизации LDAP-пользователя в Центральном Пульте не создаются новые группы;
- true – при авторизации LDAP-пользователя создаются все его группы в Центральном Пульте, пользователь автоматически добавляется в свои группы.

По умолчанию – `false` – при авторизации LDAP-пользователей в Центральном Пульте не создаются новые группы.

ldap.**login**

Логин администратора LDAP. Администратор должен иметь доступ к пользователям и группам, которые должны быть импортированы в Центральный Пульт.

ldap.**login_attribute_name**

Атрибут, в котором хранятся логины пользователей для входа в систему.

Указывается, если необходимо использовать нестандартный атрибут.

Стандартный атрибут – "**name**".

ldap.**pass**

Пароль администратора LDAP.

ldap.**suffix**

Корневой элемент (как правило, доменное имя организации).

ldap.**update_existing_groups**

Разрешает обновлять группы пользователей, созданные в Центральном Пульте, до LDAP-групп при импорте групп пользователя из LDAP и совпадении имен групп;

- false – при входе LDAP-пользователей обновления групп не происходит.
- true – при входе LDAP-пользователя его группы в Центральном Пульте, совпадающие с группами в LDAP, обновляются до LDAP-групп.

По умолчанию – `false` – при входе LDAP-пользователей обновления групп не происходит.

Для обновленной группы отключается возможность смены имени и добавляется источник – LDAP.

ldap.**url**

Адрес LDAP-сервера. Поддерживаются протоколы LDAP и LDAPs.

monitoring

В этом разделе задаются настройки пользовательских сенсоров мониторинга.

```
"monitoring": {
    "custom_tasks_path": "/opt/saymon-agent/custom_tasks"
}
```

monitoring.**custom_tasks_path**

Полный путь до папки с файлами пользовательских проверок.

mqtt

В этом разделе задаются настройки подключения к MQTT-брокеру.

```
"mqtt" : {
    "broker" : "mqtt://username:password@localhost:1883"
```

```
}
```

mqtt.**broker**

Адрес и порт брокера.

По умолчанию – `"mqtt://localhost:1883"`.

WARNING	Для аутентификации по имени пользователя и паролю нужно указать пользовательские данные перед адресом сервера.

openTsdb

В этом разделе задаются параметры доступа к OpenTSDB.

```
"openTsdb" : {
    "enabled" : true,
    "host" : "localhost",
    "port" : 4242,
    "timeout" : 15000
}
```

openTsdb.**enabled**

Запись исторических данных в OpenTSDB;

- false – запись выключена,
- true – запись включена.

По умолчанию – `true` – запись включена.

openTsdb.**host**

Адрес хоста с OpenTSDB.

По умолчанию – `"localhost"`.

openTsdb.**port**

Порт OpenTSDB.

По умолчанию – 4242.

openTsdb.**timeout**

Время на ожидание ответа от OpenTSDB (в миллисекундах).

По умолчанию – `15000` – 15 секунд.

push_notification

В этом разделе задаются параметры push-уведомлений в мобильном приложении. Для работы с ними используется Firebase Cloud Messaging (FCM).

```
"push_notification" : {
    "disabled" : true,
    "key_path" : "/etc/saymon/saymon-mobile-firebase-adminsdk.json",
    "on_state_change" : true,
    "timeout" : 0,
    "url" : "https://saymon-mobile.firebaseio.com"
}
```

push_notification.**disabled**

Выключает уведомления;

- true – уведомления выключены,
- false – уведомления включены.

По умолчанию – `true` – уведомления выключены.

push_notification.**key_path**

Путь к ключу авторизации сервера Центрального Пульта на сервере Firebase.

По умолчанию – `"/etc/saymon/saymon-mobile-firebase-adminsdk.json"`.

push_notification.**on_state_change**

Включает отправку уведомления при изменении состояний на случай другого источника уведомлений – MQTT-сообщений;

- true – отправка включена,
- false – отправка выключена.

По умолчанию – `true` – отправка включена.

push_notification.**timeout**

Время, через которое каждому пользователю отправляется уведомление (в миллисекундах).

По умолчанию – `0` – задержки нет.

push_notification.**url**

URL, полученный пользователем от Firebase, для принятия содержимого уведомлений с сервера.

По умолчанию – `"https://saymon-mobile.firebaseio.com"`.

resource_server

В этом разделе задаются параметры, связанные с хранением файлов, загруженных в Центральный Пульт.

```
"Resource_server" : {
    "debug" : false,
    "file_storage_dir" : "/var/saymon/resources",
    "ip_address" : "127.0.0.1",
    "port" : 8092
}
```

resource_server.**debug**

Включает debug-режим для логирования в файл `/var/log/saymon/saymon-server.log`;

- false – режим выключен,
- true – режим включен.

По умолчанию – `false` – режим выключен.

resource_server.**file_storage_dir**

Путь к директории для хранения документов, прикрепляемых к объектам.

По умолчанию – `"/var/saymon/resources"`.

resource_server.**ip_address**

Адрес Resource-сервера.

По умолчанию – `"127.0.0.1"`.

resource_server.**port**

Порт Resource-сервера.

По умолчанию – `8092`.

rest_server

В этом разделе задаются параметры REST-сервера.

```
"Rest_server" : {
    "base_url" : "/api",
    "colorize_log" : false,
    "debug" : false,
    "document_download_url" : "http://localhost/node/api/docs",
    "ip_address" : "127.0.0.1",
    "keepAliveTimeout" : 5000,
    "num_workers" : 1,
    "port" : 8090,
    "public_url" : "http://mycompany.com/index.html",
    "session_ttl" : 3600000,
    "snmp_mib_url" : "http://192.168.1.189:5550",
    "update_download_url" : "http://localhost/node/api/agents/update"
}
```

rest_server.**base_url**

Путь к API.

По умолчанию – `"/api"`.

rest_server.**colorize_log**

Включает цветную раскраску лога;

- false – раскраска выключена,
- true – раскраска включена.

По умолчанию – `false` – раскраска выключена.

rest_server.**debug**

Включает debug-режим для логирования в файл `/var/log/saymon/saymon-server.log`;

- false – режим выключен,
- true – режим включен.

По умолчанию – `false` – режим выключен.

rest_server.**document_download_url**

URL к файлам, сохраненным в **$document_storage_dir**.

По умолчанию – `"http://localhost/node/api/docs"`.

rest_server.**ip_address**

Адрес хоста для запуска REST-сервера.

По умолчанию – `"127.0.0.1"`.

rest_server.**keepAliveTimeout**

Время ожидания сервером следующего запроса до разрыва текущего соединения (в миллисекундах).

По умолчанию – `5000` – 5 секунд.

rest_server.**num_workers**

Число процессов для загрузки данных.

По умолчанию – `1`.

rest_server.**port**

Порт REST-сервера.

По умолчанию – `8090`.

rest_server.**public_url**

Адрес для доступа к web-интерфейсу из уведомлений.

rest_server.**session_ttl**

Время, по истечении которого происходит автоматическое завершение сессии при неактивности пользователя (в миллисекундах).

По умолчанию – `3600000` – 1 час.

rest_server.**snmp_mib_url**

Адрес сервера трансляции SNMP OID в текстовый формат.

rest_server.**update_download_url**

Путь к файлу для обновления агента.

По умолчанию – `"http://localhost/node/api/agents/update"`.

rest_server.**limits**

В этом подразделе задаются ограничения на число запросов к REST API.

```
"limits" : [
    {
        "__comment": "администратора не ограничиваем",
        "user": "5048c849d7b6e40593dfee71"
    },
    {
        "__comment": "группу дежурной смены не ограничиваем",
        "group": "60e591928196105ec9c74636"
    },
    {
        "__comment": "запросы на states и classes не ограничиваем",
        "uri": ["/states", "/classes"]
    },
    {
        "__comment": "ограничиваем запросы на объекты",
        "uri": ["/objects"],
        "rate": "100/minute"
    },
    {
        "__comment": "ограничиваем авторизовавшихся пользователей с 127.0.0.1",
        "user": true,
        "ip": '127.0.0.1',
        "rate": "120/minute"
    },
    {
        "__comment": "ограничения для неавторизованных пользователей",
        "user": false,
        "rate": "10/minute"
    }
]
```

По умолчанию ограничения не накладываются.

WARNING	Для каждого запроса поиск правил осуществляется сверху вниз. Применяется только одно правило.

rest_server.limits.**group**

Группы пользователей, для которых устанавливается ограничение;

- string\|string[] – ID группы или групп, к которым применяется правило.

Примеры:

```
"limits": [
    {
        "group": "60e562277936944ebf158282",
        ...
    }
```

```
]
```

```
"limits": [
    {
        "group": [
            "60e562277936944ebf158282",
            "616406e1a3db5948cd815438"
        ],
        ...
    }
]
```

rest_server.limits.ip

IP-адреса, для которых устанавливается ограничение;

- true – свой лимит для каждого IP-адреса;
- string\|string[] – IP-адрес или список адресов, к которым применяется правило.

Примеры:

```
"limits": [
    {
        "ip": true,
        ...
    }
]
```

```
"limits": [
    {
        "ip": "127.0.0.1",
        ...
    }
]
```

```
"limits": [
    {
        "ip": [ "127.0.0.1", "192.168.1.2" ],
        ...
    }
]
```

rest_server.limits.rate

Количество запросов в единицу времени. Если не указан, то ограничение не накладывается.

Задаётся в формате **количество_запросов/единица_измерения_времени**.

Возможные единицы измерения времени:

- y – год,
- Q – квартал,
- M – месяц,
- w – неделя,
- d – день,
- h – час,
- m/minute – минута,
- s – секунда,
- ms – миллисекунда.

Пример:

```
"limits": [
    {
        "rate": 100/minute,
        ...
    }
]
```

rest_server.limits.**uri**

REST-методы, к которым устанавливается ограничение;

- true – свой лимит для каждого метода;
- string\|string[] – метод или список методов, к которым применяется правило.

В качестве пути могут быть заданы любые API без приставки /node/api, например, "**/time**".

Примеры:

```
"limits": [
    {
        "uri": true,
        ...
    }
]
```

```
"limits": [
    {
```

```
        "uri": "/ping",
        ...
    }
]
```

```
"limits": [
    {
        "uri": [
            "/ping",
            "/node/api/objects/:id/stat"
        ],
    ...
    }
]
```

rest_server.limits.user

Пользователи, для которых устанавливается ограничение;

- true – свой лимит для каждого авторизованного пользователя;
- false – лимит по IP-адресу для каждого авторизованного пользователя;
- string\|string[] – ID пользователя или пользователей, к которым применяется правило.

Примеры:

```
"limits": [
    {
        "user": true,
        ...
    }
]
```

```
"limits": [
    {
        "user": false,
        ...
    }
]
```

```
"limits": [
    {
        "user": "6048c849d7b6e40593dfee71",
        ...
    }
]
```

```
"limits": [
    {
        "user": [
            "57ff6853fa6db3a63d16d07b",
            "6048c849d7b6e40593dfee71"
        ],
    ...
    }
]
```

server

В этом разделе задаются общие параметры сервера.

server.**agent_registration_parent_id**

ID объекта, в котором появляются агенты при саморегистрации.

По умолчанию – `"1"`.

server.**analytics_enabled**

Включает аналитику значений метрик в процесс обработки данных;

- false – аналитика выключена,
- true – аналитика включена.

По умолчанию – `false` – аналитика выключена.

server.**analytics_processes**

Количество логических ядер, выделяемых для обработки аналитики временных рядов.

По умолчанию используются все доступные ядра.

server.**colorize_log**

Включает цветную раскраску лога;

- false – раскраска выключена,
- true – раскраска включена.

По умолчанию – `false` – раскраска выключена.

server.**comet_ping_interval**

Временной интервал между отправками Comet-сервером сообщений (в миллисекундах).

По умолчанию – `5000` – 5 секунд.

server.**comet_ping_timeout**

Время ожидания сообщения от Comet-сервера (в миллисекундах).

По умолчанию – `12000` – 12 секунд.

server.**comet_port**

Порт для соединения.

По умолчанию – `8091`.

server.**comet_secure**

Включает SSL-соединение;

- false – соединение выключено,
- true – соединение включено.

По умолчанию – `false` – соединение выключено.

server.**comet_ssl_certificate**

Путь к сертификату.

server.**comet_ssl_key**

Путь к ключу.

server.**conditional_incidents_enabled**

Включает функционал генерации аварий;

- false – функционал выключен,
- true – функционал включен.

По умолчанию – `false` – функционал выключен.

server.**conditional_incidents_script**

Путь до скрипта, выполняющегося при возникновении аварии.

WARNING	Применяется только при включенном функционале генерации аварий: `server.conditional_incidents_enabled = true`

server.**debug**

Включает debug-режим для логирования в файл `/var/log/saymon/saymon-server.log`;

- false – режим выключен,
- true – режим включен.

По умолчанию – `false` – режим выключен.

server.**default_result_timeout**

Время, через которое срабатывает условие **Нет данных от объекта** с момента создания объекта или получения последних данных (в миллисекундах).

По умолчанию – `120000` – 2 минуты.

server.**default_state_id**

Состояние объекта по умолчанию.

По умолчанию – `7` – **NO DATA**.

server.**discovery_parent_id**

ID объекта, в котором появляются найденные агентами сетевые устройства.

По умолчанию – `"1"`.

server.**event_log_max_bytes**

Объем Журнала Событий в MongoDB, при достижении которого происходит ротация данных (в байтах).

По умолчанию – `"1 G"` – 1 гигабайт.

server.**history_temporary_storage_period**

Интервал времени для буферизации метрик, по истечении которого все данные из Redis записываются в OpenTSDB (в миллисекундах).

Применяется, только если **history_update_period** равен **0**.

server.**history_update_period**

Интервал записи исторических данных (в миллисекундах).

0 – немедленная запись пришедших значений.

По умолчанию – `60000` – 1 минута.

server.**history_write_length**

Ограничение количества точек в одном запросе к OpenTSDB.

При указании этого параметра данные в базу записываются по частям.

TIP	Функционал полезен при превышении размера записываемых за один раз данных. Рекомендуемое значение в таком случае – 100000.

По умолчанию ограничения нет, запись данных в OpenTSDB происходит без разбиения на части.

server.**incident_timeout**

Время перехода аварии из списка активных аварий в список исторических аварий после погашения аварии (в миллисекундах).

По умолчанию – `120000` – 2 минуты.

server.**notification_buffering_period**

Период ожидания для сбора сообщений о смене состояний объектов и отправки группового уведомления (в миллисекундах).

По умолчанию – `0` – буферизация отключена.

server.**retain_expired_stat**

Включает хранение последних полученных данных после их устаревания;

- false – хранение выключено,
- true – хранение включено.

По умолчанию – `false` – хранение выключено.

server.**script_trigger_timeout**

Максимальное время выполнения триггера или операции (в миллисекундах).

По умолчанию – `30000` – 30 секунд.

server.**self_object_id**

ID объекта, используемого для самоконтроля сервера.

server.**self_monitoring_period**

Период между процессами самоконтроля сервера.

По умолчанию – 2 минуты.

server.**sms_script**

Путь до скрипта, отправляющего SMS-уведомления.

server.**stat_local_timestamp_field_name**

Имя поля, где передается время, с которым нужно сохранять данные в OpenTSDB.

По умолчанию – `"localTimestamp"`.

server.**stat_scan_period**

Период проверки актуальности пришедших данных (в миллисекундах).

По умолчанию – `3000` – 3 секунды.

server.**use_stat_timestamp**

Сохраняет временную метку (поле **timestamp**) из данных агента;

- false – значение поля **timestamp** из данных агента заменяется на временную метку сервера,
- true – используется значение поля **timestamp** из данных агента, оно не перезаписывается временем сервера.

По умолчанию – `false` – временная метка из данных агента перезаписывается временем сервера.

server.**voice_call_script**

Путь до скрипта, осуществляющего голосовой вызов.

server.**email**

В этом подразделе задаются параметры доступа к почтовому серверу:

```
"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
    },
}
```

server.email.**disabled**

Выключает отправку почтовых уведомлений;

- true – отправка выключена,
- false – отправка включена.

По умолчанию – `true` – отправка выключена.

server.email.**max_json_length**

Ограничение размера письма с уведомлением (в символах).

По умолчанию – `1000` – 1000 символов.

server.email.**fields**

Данные об отправителе уведомлений.

server.email.fields.**from**

Почтовый адрес отправителя.

server.email.**transport**

Данные почтового сервера.

server.email.transport.**host**

Адрес почтового сервера.

server.email.transport.**port**

Порт почтового сервера.

server.email.transport.**secure**

Включает использование TLS при подключении к серверу;

- false – использоване TLS выключено,
- true – использоване TLS включено.

Значение по умолчанию зависит от порта.

server.email.transport.**service**

Встроенный в коннектор набор служб.

При наличии задаёт **host**, **port**, **secure** автоматически.

По умолчаниию – "**Gmail**".

server.email.transport.**auth**

Данные для аутентификации пользователя.

server.email.transport.auth.**pass**

Пароль пользователя.

server.email.transport.auth.**user**

Логин пользователя.

server.**user**

В этом подразделе задаются параметры пользователей:

```
"user" : {
    "auth_enabled": true,
    "new_user_access": "all",
    "template": {
        "permissions": ["manage-objects", "view-section-stat"],
        "objectPermissions": {
            "include": [],
            "exclude": ["5fb643ddf277b96c8401119b", "5f8dc28407e86603bfe281f8"]
        }
    },
    "usersRoot": "5800d9aaac7bf0f90d3d520e"
}
```

server.user.**auth_attempts**

Настройки блокировки пользователя при неудачной попытке входа.

При определённом количестве неудачных попыток входа, указанном в параметре `attempts` пользователь блокируется. Все запросы от этого пользователя возвращают код `429 Too Many Requests`.

При успешном логине или через определённое количество времени, указанное в

параметре `block_period`, количество неудачных попыток входа сбрасывается. Если пользователь уже был заблокирован, то он будет разблокирован через время, указанное в `block_period`.

```
"user" : {
    ...
    "auth_attempts": {
        "enabled": true,
        "attempts": 3,
        "block_period": 60000
    },
    ...
}
```

server.user.auth_attempts.**enabled**

Включает функцию блокировки пользователя при неудачных попытках входа.

- false – блокировка выключена,
- true – блокировка включена.

По умолчанию – `true` – блокировка включена.

server.user.auth_attempts.**attempts**

Количество неудачных попыток, которое приведёт блокировке.

По умолчанию – 3.

server.user.auth_attempts.**block_period**

Период сброса ключа неудачных попыток входа в миллисекундах.

Если пользователь уже был заблокирован за неудачные попытки входа, то он будет разблокирован через период, указанный в этом параметре.

По умолчанию – `300000`.

server.user.**auth_enabled**

Включает самостоятельную регистрацию и возможность удалить собственную учётную запись для пользователей;

- false – регистрация и удаление выключены,
- true – регистрация и удаление включены.

По умолчанию – `false` – регистрация и удаление выключены.

server.user.**auth_token_enabled**

Включает аутентификацию с помощью токена API.

- false – аутентификация по токену выключена,
- true – аутентификация по токену включена.

По умолчанию – `true` – аутентификация по токену включена.

server.user.**lang_default**

Язык пользователей по умолчанию;

- "en" – английский,
- "it" – итальянский,
- "ru" – русский.

По умолчанию – `"en"` – английский.

server.user.**new_user_access**

Права доступа к объектам для нового пользователя;

- "all" – есть доступ ко всем объектам,
- "not" – нет доступа ни к одному объекту.

По умолчанию – `"all"` – доступ ко всем объектам.

server.user.**usersRoot**

Идентификатор корневого объекта для создаваемых новым пользователем объектов.

По умолчанию – `"1"`.

server.user.**template**

Шаблон прав нового пользователя.

WARNING	Данные параметры применяются только для пользователей, регистрирующихся самостоятельно.

server.user.template.**permissions**

Список прав на операции, доступных пользователю по умолчанию.

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.

snmp

В этом разделе задаются параметры обработки SNMP-трапов.

```
"snmp": {
    "filter": "/users/admin/tmp/snmp-filter.js"
}
```

snmp.**filter**

Путь к файлу с функцией фильтрации входящих SNMP Trap.

Если функция возвращает значение `false`, сообщение отбрасывается.

Приведённый ниже пример блокирует трапы, которые содержат любую из строк, указанных в массиве *filters*:

Пример функции фильтрации

```
module.exports = function(message, rawMessage) {
    rawMessage = String(rawMessage);
    //console.log(message, rawMessage);
    let filters = [ 'HLR Call Processing Message',
        'AIR Rejected - No EPS Subscription',
        'GPRS SIBB Processing Exception',
        'AuC Call Processing Message',
        'Received notification from CLIM',
        'AuC Event Log Clean Up',
        'SOFT 0006'
    ];
    return !filters.some((filter) => rawMessage.includes(filter));
}
```

В данном примере блокируется трап с *enterpriseOid = '.1.3.6.1.4.1.169.10.1.1' И genericTrap = 6*

И specificTrap = 5:

Пример функции фильтрации

```
module.exports = function(message, rawMessage) {
    return message.payload.enterpriseOid != '.1.3.6.1.4.1.169.10.1.1' &&
message.payload.genericTrap != 6 && message.payload.specificTrap != 5;
}
```

state_history_log

```
"state_history_log": {
    "max_time_range_interval": 30
}
```

state_history_log.**max_time_range_interval**

Определяет максимальное значение временного интервала, которое может указать пользователь при запросе отчёта с указанием параметров, которые замедляют фильтрацию истории состояний (`classId`, `tagId`, `entityName`). Указывается в днях.

WARNING	Если указать значение `0`, то пользователь не будет ограничен в выборе временного диапазона, но это может привести к долгой обработке запросов при большом количестве объектов.

По умолчанию – `90` дней.

wetty

В этом разделе задаются параметры логирования истории взаимодействия пользователей с системой через терминал Wetty.

NOTE	История каждой пользовательской сессии сохраняется в отдельный файл. Формат имени файла: **YYYY-MM-DD_HH_MM_SS_PROTOCOL_IP_USERNAME.log**, например **2023-01-12_11-56-17_ssh_127.0.0.1_admin.log**.

WARNING	Если используется версия Центрального Пульта ниже 3.12.86, при переходе на новую версию необходимо:
	
	1. Внести изменения в файл `/etc/nginx/sites-available/saymon`:
	
	`location ~ /wetty/(.*)`
	`{`
	`  # take path and query after wetty/ and append it to`

```
proxy_pass
  proxy_pass http://127.0.0.1:3001/wetty/$1?$query_string;
  proxy_http_version 1.1;
  proxy_set_header Upgrade $http_upgrade;
  proxy_set_header Connection "upgrade";
}
```

2. Перезапустить nginx:

```
$ service nginx reload
```

```
"wetty" : {
    "log_dir" : "/tmp",
    "raw_mode": true
}
```

wetty.**log_dir**

Путь к папке для хранения log-файлов.

wetty.**raw_mode**

Включает режим записи логов с управляющими ANSI-символами;

- true – логи записываются с управляющими ANSI-символами;
- false – логи записываются без управляющих ANSI-символов.

По умолчанию – `true` – логи записываются с ANSI-символами.

zabbix

В этом разделе задаются параметры для импорта данных из системы Zabbix.

WARNING	По умолчанию сервер использует систему подключения к Zabbix 6.0. Для использования Zabbix 5.x, нужно выставить параметр zabbix5 в `true`.

```
"zabbix" : [
    {
        "url" : "http://192.168.1.215/zabbix/api_jsonrpc.php",
        "user" : "saymon",
        "password" : "saymon_user_password",
        "parent_id" : "58b586d5c3a2f96642e25537",
        "debug": true,
        "polling_period": 30000,
        "classes" : {
```

```
            "4" : "CPU",
            "619503be0ffb595aebe22222" : "Memory

General"
        }
    }
]
```

zabbix.classes

Массив соответствий идентификаторов классов и имён обнаруженных объектов. Объектам с указанными именами будут при создании присвоены соответствующие классы.

zabbix.debug

Включает вывод параметров и результатов запросов в лог;

- false – вывод выключен,
- true – вывод включен.

По умолчанию – `false` – вывод выключен.

zabbix.parent_id

ID объекта, используемого в качестве родителя для объектов, импортируемых из Zabbix.

zabbix.password

Пароль пользователя Zabbix с правами на чтение хостов или групп хостов, данные о которых необходимо импортировать.

zabbix.polling_period

Период обновления данных (в миллисекундах).

По умолчанию – `120000` – 2 минуты.

zabbix.url

Адрес файла `api_jsonrpc.php` требуемой инсталляции Zabbix.

zabbix.user

Имя пользователя Zabbix с правами на чтение хостов или групп хостов, данные о которых необходимо импортировать.

zabbix.zabbix5

Использовать систему подлючения для Zabbix 5.x.

- `true` – сервер использует систему для Zabbix 5.x,
- `false` – сервер использует систему для Zabbix 6.0.

По умолчанию – false – Используется система подключения для Zabbix 6.0.

NOTE	В версиях Центрального пульта старше 3.14.89 этот параметр не требуется, так как в них нет поддержки Zabbix 6.0.

zabbix.request_options

В этом подразделе можно задать изменения опций для запроса хостов, триггеров и данных.

```
"request_options": {
    "hosts": {
        "filter": {
            ...
        }
    },
    "triggers": {
        "filter": {
            ...
        }
    },
    "data": {
        "filter": {
            ...
        }
    }
}
```

При интеграции Центрального пульта версии 3.14.89 и выше с Zabbix 6.0 дополнительно можно задать изменения опций для тегов:

```
"request_options": {
    ...
    "tags" [
        {
            "tag" : "ZBXHEALTH"
        }
    ],
    ...
}
```

zabbix.request_options.hosts

Параметры для запроса хостов.

zabbix.request_options.hosts.filter

Поля фильтров для запроса хостов.

Все возможные поля фильтров hosts представлены в документации по Zabbix API.

zabbix.request_options.triggers

Параметры для запроса триггеров.

zabbix.request_options.triggers.filter

Поля фильтров для запроса триггеров.

Все возможные поля фильтров triggers представлены в документации по Zabbix API.

zabbix.request_options.data

Параметры для запроса данных.

zabbix.request_options.data.filter

Поля фильтров для запроса данных.

Все возможные поля фильтров data представлены в документации по Zabbix API.

Приложение Г (обязательное)

Конфигурация Web-приложения

Параметры конфигурации web-приложения Центрального пульта хранятся в конфигурационном файле `/etc/saymon/saymon-client.yaml`.

WARNING	Файлы с параметрами конфигурации web-приложения могут храниться в следующих файлах (в порядке снижения приоритета): • MongoDB – параметры конфигурации, которые можно изменить через REST API; • `/etc/saymon/saymon-client.yaml` – параметры конфигурации web-приложения в формате YAML (основной файл конфигурации); • `/etc/saymon/saymon-client.json` – параметры конфигурации web-приложения в формате JSON (дополнительный файл

конфигурации);

- `/etc/saymon/client-config.js` – параметры конфигурации web-приложения в старом формате JS (используется со старыми версиями web-приложения);
- `./backend/server/saymon-client.yaml` – параметры конфигурации web-приложения по умолчанию.

Файл конфигурации с низким приоритетом используется в том случае, если отсутствует файл конфигурации с высоким приоритетом.

agentInstallerUrl

Ссылка для скачивания установочного скрипта для агента.

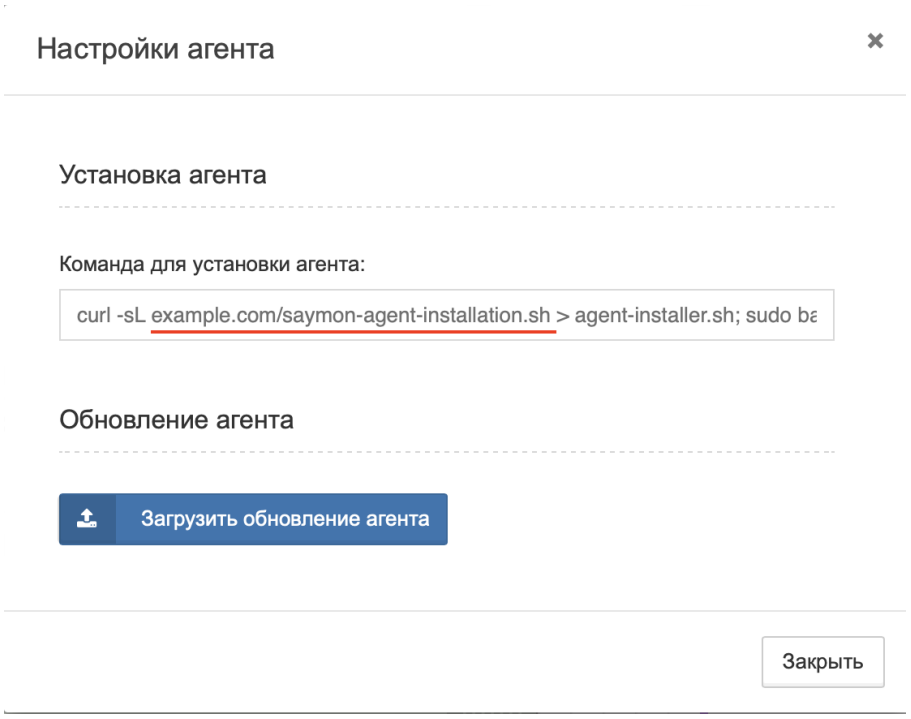
YAML

```yaml
agentInstallerUrl: example.com/saymon-agent-installation.sh
```

JSON

```
{
  "agentInstallerUrl": "example.com/saymon-agent-installation.sh",
  ...
}
```

Ссылка добавляется в сгенерированную команду для установки агента при автоинсталляции.



По умолчанию ссылка не задана.

authEnabled

Включает самостоятельную регистрацию для пользователей;

YAML

```yaml
authEnabled: false
```

JSON

```
{
  "authEnabled": false,
  ...
}
```

- false – регистрация выключена,
- true – регистрация включена.

По умолчанию – **false** – регистрация выключена.

authTokenEnabled

Настраивает отображение поля **Ссылка для авторизации** и кнопки создания токена API в окне настроек пользователя.

YAML

```yaml
authTokenEnabled: false
```

JSON

```
{
  "authTokenEnabled": false,
  ...
}
```

- false – поле выключено,
- true – поле включено.

По умолчанию – **true** – поле включено.

authentication

В этой секции можно настроить параметры аутентификации пользователей.

YAML

```yaml
authentication:
  availableMethods:
    - native
    - keycloak
  methodByDefault: keycloak
  services:
    keycloak:
      configUri: keycloak_frontend.json
      legacySupport: false
```

JSON

```json
{
  "authentication": {
    "availableMethods": [
      "native",
      "keycloak"
    ],
    "methodByDefault": "keycloak",
    "services": {
      "keycloak": {
        "configUri": "keycloak_frontend.json",
        "legacySupport": false
      }
    }
  }
}
```

authentication.**availableMethods**

Список доступных методов аутентификации:

- **native** – аутентификация по логину и паролю пользователя;
- **keycloak** – аутентификация через реквизиты учетной записи пользователя в Keycloak.

По умолчанию – **native**.

authentication.**methodByDefault**

Метод аутентификации, который используется по умолчанию.

По умолчанию – **native**.

authentication.**services**

Секция с настройками сервисов аутентификации.

authentication.services.**keycloak**

Секция с настройками аутентификации через реквизиты учетной записи в Keycloak.

authentication.services.keycloak.**configUri**

Имя файла настроек соединения, сгенерированного Keycloak-сервером для web-приложения.

Файл настроек соединения необходимо разместить на сервере Центрального Пульта в каталоге `/etc/saymon`.

authentication.services.keycloak.**legacySupport**

Включает поддержку работы с Keycloak-сервером версии ниже 19;

- false – поддержка работы с Keycloak-сервером версии ниже 19 выключена;
- true – поддержка работы с Keycloak-сервером версии ниже 19 включена.

По умолчанию – **false** (работа с Keycloak-сервером версии ниже 19 не поддерживается).

canEditObjectStyle

Включает возможность вручную настраивать CSS-стили для объектов и связей.

YAML

```yaml
canEditObjectStyle: true
```

JSON

```json
{
  "canEditObjectStyle": true,
  ...
}
```

При включении опции в разделе **Параметры** появляется подраздел **Стили** (объектов, связей);

NOTE	При выключении опции, применённые стили не убираются с объектов и связей.

- false – настройка стилей выключена,
- true – настройка стилей включена.

По умолчанию – **true** – настройка стилей включена.

comet

Секция с настройками взаимодействия web-приложения с сервером Центрального Пульта через Comet-сервер.

YAML

```yaml
comet:
  port: 8091
  url:
    - "https://cpult.ru/ws"
    - "http://10.78.5.11:8092"
```

JSON

```json
{
  "comet": {
    "port": 8091,
    "url": [
      "https://cpult.ru/ws",
      "http://10.78.5.11:8092"
    ]
  }
}
```

comet.**port**

Номер порта, по которому web-приложение подключается к Comet-серверу.

По умолчанию используется порт **8091**.

comet.**url**

Адреса для подключения web-приложения к Comet-серверу из разных сетей. При задании адреса можно использовать IP-адрес, доменное имя или имя хоста. Также дополнительно можно указывать порт.

Если задано несколько адресов, то при установке подключения осуществляется их перебор до успешной установки соединения. Адрес выбирается случайным образом.

WARNING	Если задан параметр *comet.url*, то параметр *comet.port* игнорируется.

configuration

Секция с настройками окна конфигурации системы.

YAML

```yaml
configuration:
  objectPermissions:
    displayType: tree
```

JSON

```json
{
  "configuration": {
    "objectPermissions": {
      "displayType": "tree"
    }
  }
}
```

configuration.**objectPermissions**

Настройки доступа к объектам.

configuration.objectPermissions.**displayType**

Настройки вида отображения списка объектов.

- `tree` – интерфейс управления правами в виде дерева всех объектов в системе,
- `simple` – упрощённый интерфейс управления правами доступа к объектам.

По умолчанию – `tree`.

collapseSections

Задаёт настройки первоначального отображения секций и графиков;

YAML

```yaml
collapseSections: true
```

JSON

```json
{
  "collapseSections": true,
  ...
```

```
}
```

- false – секции закреплены, графики раскрыты,
- true – секции откреплены, графики свёрнуты.

По умолчанию – **true** – секции откреплены, графики свёрнуты.

defaultClassImage

Позволяет задать стандартное изображение и его размер для объектов определенного класса.

YAML

```yaml
defaultClassImage:
  "29":
    img: /images/cloud.svg
    dim:
      - 300
      - 200
```

JSON

```json
{
  "defaultClassImage": {
    "29": {
      "img": "/images/cloud.svg",
      "dim": [
        156,
        104
      ]
    }
  }
}
```

defaultClassImage.**<number>**

Идентификатор класса, например, "**29**".

defaultClassImage.<number>.**img**

Путь до изображения относительно папки `/usr/local/saymon/client/`.

WARNING	Стандартным изображением может быть только векторный файл в формате SVG. Изображения других форматов (например PNG, JPEG или

	GIF) будут отображаться в окне настройки класса, но не будут отображаться в окне параметров и стандартном виде.

defaultClassImage.<number>.**dim**

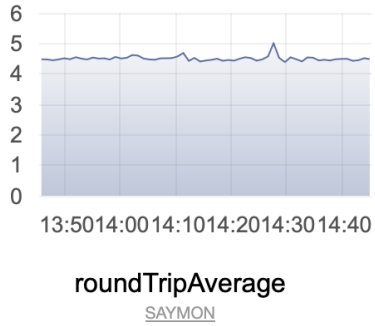
Размер изображения (в пикселях, [ширина, высота]).

По умолчанию – `[300, 200]`

defaultEmbeddableLink

Настройка ссылки под встраиваемым виджетом, которая генерируется по умолчанию. При встраивании, эту ссылку можно изменить.

Этот раздел позволяет настроить текст и URL ссылки под виджетом:



YAML

```yaml
defaultEmbeddableLink:
  url: http://cpult.ru/
  name: Центральный Пульт
```

JSON

```json
{
  "defaultEmbeddableLink": {
    "url": "http://cpult.ru/",
    "name": "Центральный Пульт"
  },
}
```

defaultEmbeddableLink.**url**

URL ссылки под виджетом.

defaultEmbeddableLink.name

Текст ссылки под виджетом.

defaultPieChartColors

Цвета секторов по умолчанию для круговой и радиальной диаграмм. Если секторов больше чем цветов по умолчанию, цвета начинают повторяться сначала.

YAML

```yaml
defaultPieChartColors:
  - "#a90329"
  - "#3276b1"
  - "#009F3D"
  - "#9868B8"
  - "#F2C001"
  - "#FF742C"
  - "#90574E"
  - "#EB77BE"
  - "#FF9796"
  - "#AAC6E5"
  - "#8BDE90"
  - "#FDB5D0"
```

JSON

```json
{
  "defaultPieChartColors": [
    "#a90329",
    "#3276b1",
    "#009F3D",
    "#9868B8",
    "#F2C001",
    "#FF742C",
    "#90574E",
    "#EB77BE",
    "#FF9796",
    "#AAC6E5",
    "#8BDE90",
    "#FDB5D0"
  ]
}
```

disabledSections

Позволяет удалить секции, которые показываются при подробном отображении объекта в верхней части экрана.

YAML

```yaml
disabledSections:
  - audit-log
  - config-log
  ...
```

JSON

```json
{
  "disabledSections": [
    "audit-log",
    "config-log",
    ...
  ]
}
```

Список всех доступных секций:

- audit-log (История изменений),
- config-log (Изменения конфигурации),
- documents (Документы),
- entity-incident-conditions (Условия генерации аварий),
- entity-settings (Параметры объектов, связей),
- entity-state-conditions (Условия перехода состояний),
- history-graph (Графики),
- monitoring (Мониторинг),
- operations (Операции),
- properties (Свойства),
- stat (Данные),
- stat-rules (Правила формирования данных),
- state-history (История состояний),
- state-triggers (Действия при смене состояний),
- widgets (Виджеты).

По умолчанию – пустой список, все секции включены.

documentation

В этом разделе можно настроить ссылки на различную документацию для Центрального

Пульта.

YAML

```yaml
documentation:
  main:
    root: https://docs.saymon.tech
    ru: https://docs.cpult.ru
  notificationTemplates:
    root: https://docs.saymon.tech/web-interface/system-configuration-window/notification-templates.html
    ru: https://docs.cpult.ru/web-interface/system-configuration-window/notification-templates.html
  restApi:
    root: https://api.cpult.ru
```

JSON

```json
"documentation": {
  "main": {
      "root": "https://docs.saymon.tech",
      "ru": "https://docs.cpult.ru"
  },
  "notificationTemplates": {
      "root": "https://docs.saymon.tech/web-interface/system-configuration-window/notification-templates.html",
      "ru": "https://docs.cpult.ru/web-interface/system-configuration-window/notification-templates.html"
  },
  "restApi": {
      "root": "https://api.cpult.ru"
  }
}
```

TIP

Ссылки, которые не начинаются с `http://` или `https://` относительны от адреса сервера Центрального Пульта.

Например, следующая конфигурация откроет документацию по адресу `http://<saymon-hostname>/documentation/ru`:

```yaml
documentation:
  main:
    root: documentation/ru
```

Для каждого поддерживаемого языка можно задать свою ссылку:

- `root/en` – английский язык;

- `ru` – русский язык;
- `it` – итальянский язык.

Например, конфигурация ссылок на документацию REST API для каждого поддерживаемого языка может выглядеть следующим образом:

```
documentation:
  restApi:
    en: https://docs.saymon.tech
    ru: https://example.com/docs/ru-RU
    it: documentation/Italiano # https://<saymon-
hostname>/documentation/Italiano
```

documentation.**main**

Ссылки на пользовательскую документацию.

По умолчанию:

- https://docs.saymon.tech (английский и итальянский);
- https://docs.cpult.ru (русский).

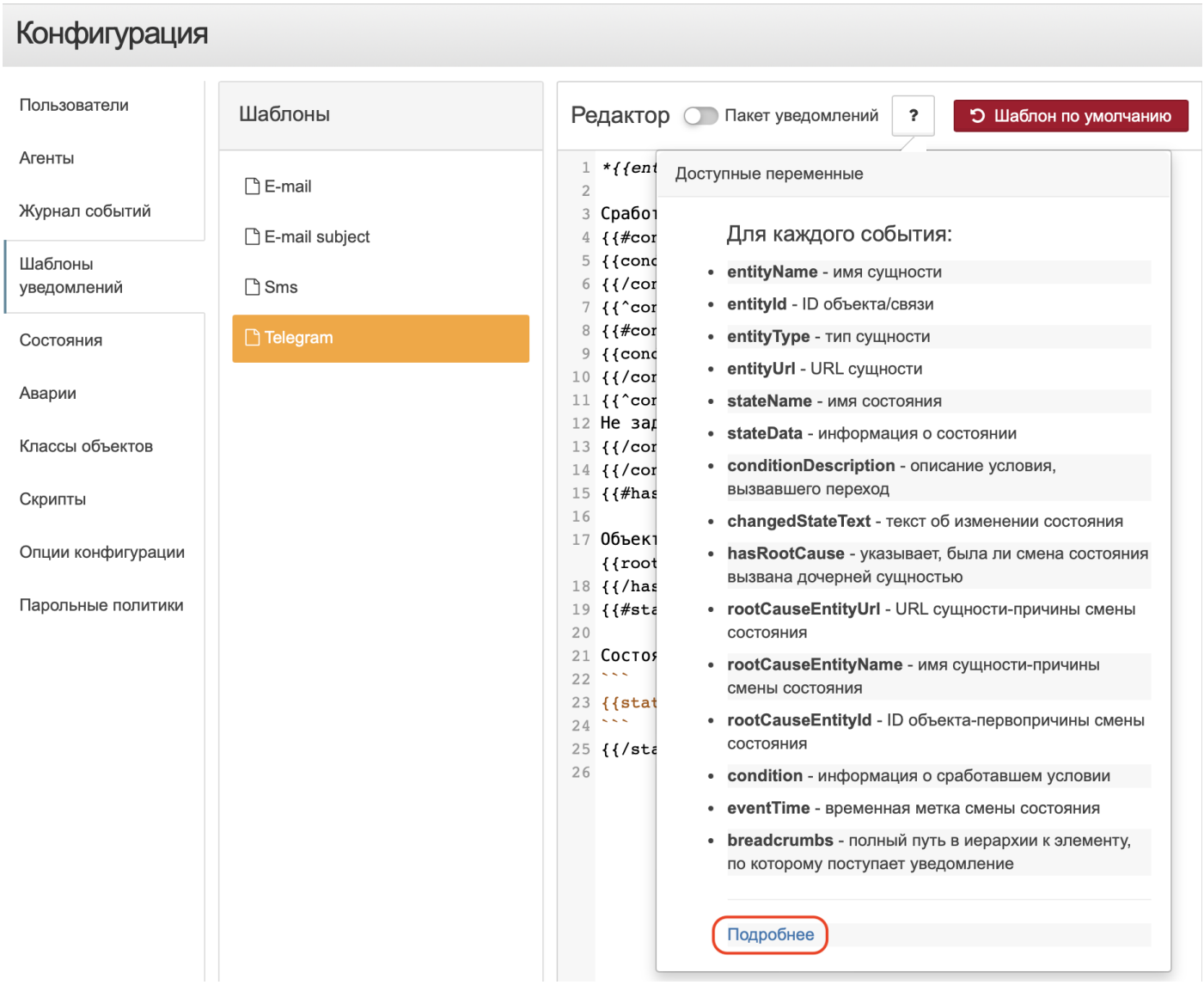
documentation.**restApi**

Ссылки на документацию по REST API.

По умолчанию (для всех языков) – https://api.cpult.ru

documentation.**notificationTemplates**

Ссылка на описание шаблонов уведомлений. Документация открывается по нажатию на кнопку "Подробнее" в настройках шаблонов уведомлений:



По умолчанию – ссылка на статью XREF_TEXT_NOT_FOUND.

enableAnalytics

Включает функционал модуля аналитики;

YAML

```yaml
enableAnalytics: false
```

JSON

```json
{
  "enableAnalytics": false,
  ...
}
```

- false – функционал выключен,

- true – функционал включен.

По умолчанию – **false** – функционал выключен.

enableJiraTrigger

Включает возможность создать задачу в Jira при смене состояния объекта.

YAML

```yaml
enableJiraTrigger: false
```

JSON

```
{
  "enableJiraTrigger": false,
  ...
}
```

- false – функционал выключен,
- true – функционал включен.

По умолчанию – **false** – функционал выключен.

enableConditionalIncidents

Включает функционал генерации аварий;

YAML

```yaml
enableConditionalIncidents: false
```

JSON

```
{
  "enableConditionalIncidents": false,
  ...
}
```

- false – функционал выключен,
- true – функционал включен.

По умолчанию – **false** – функционал выключен.

enableSmsTrigger

Включает использование SMS-уведомлений;

YAML

```yaml
enableSmsTrigger: false
```

JSON

```
{
  "enableSmsTrigger": false,
  ...
}
```

- false – SMS-уведомления выключены,
- true – SMS-уведомления включены.

По умолчанию – **false** – SMS-уведомления выключены.

enableSnmpTranslate

Включает трансляцию SNMP OID в текстовый формат;

YAML

```yaml
enableSnmpTranslate: false
```

JSON

```
{
  "enableSnmpTranslate": false,
  ...
}
```

- false – трансляция выключена,
- true – трансляция включена.

По умолчанию – **false** – трансляция выключена.

enableVoiceCallTrigger

Включает использование голосовых вызовов;

YAML

```yaml
enableVoiceCallTrigger: false
```

JSON

```json
{
  "enableVoiceCallTrigger": false,
  ...
}
```

- false – голосовые вызовы выключены,
- true – голосовые вызовы включены.

По умолчанию – **false** – голосовые вызовы выключены.

forceEmptyStandardView

Список классов для которых пользовательский интерфейс будет по умолчанию открывать стандартный вид, даже при отсутствии дочерних объектов.

YAML

```yaml
forceEmptyStandardView:
  - 1
  - 3
  - 13
  - 43
```

JSON

```json
{
  "forceEmptyStandardView": [
    1,
    3,
    13,
    43
  ]
}
```

При выборе объекта, в том числе не содержащего вложенных объектов, Центральный Пульт по умолчанию показывает для него подробную информацию.

В параметре задаётся список идентификаторов классов, при выборе пустых объектов которых Центральный Пульт переходит в стандартный вид.

По умолчанию:

- 1 (Root),
- 3 (Host),
- 13 (Node),
- 43 (Dashboard).

NOTE	Система запоминает последний выбранный вид отображения для каждого объекта в локальном хранилище браузера. Если этот вид отличается от стандартного, то система выбирает последний выбранный вид.

geoMap

Центральный Пульт позволяет привязывать объекты к их геоположению и просматривать их на карте.

YAML

```yaml
geoMap:
  serverName: openstreetmap.org
  initialPosition:
    - 30.341306
    - 59.930089
  initialZoom: 8
```

JSON

```json
{
  "geoMap": {
      "serverName": "openstreetmap.org",
      "initialPosition": [
          30.341306,
          59.930089
      ],
      "initialZoom": 8
  }
}
```

geoMap.**initialPosition**

Начальная позиция на карте в формате [долгота, широта].

Позиция указывается как массив из двух значений, где первое значение это широта, а второе – долгота:

YAML

```yaml
initialPosition:
  - 30.341306
  - 59.930089
```

JSON

```
"initialPosition": [
    30.341306,
    59.930089
],
```

По умолчанию – [30.341306,59.930089].

WARNING	Если указать данные, выходящие за пределы допустимых значений ([0,90] для широты, [0,180] для долготы), то произойдёт ошибка и карта не будет отображаться.

geoMap.**initialZoom**

Начальный масштаб карты. Допустимые значения - от 2 до 20. При выходе за эти границы, значение ограничивается допустимыми значениями.

По умолчанию – **8**.

geoMap.**serverName**

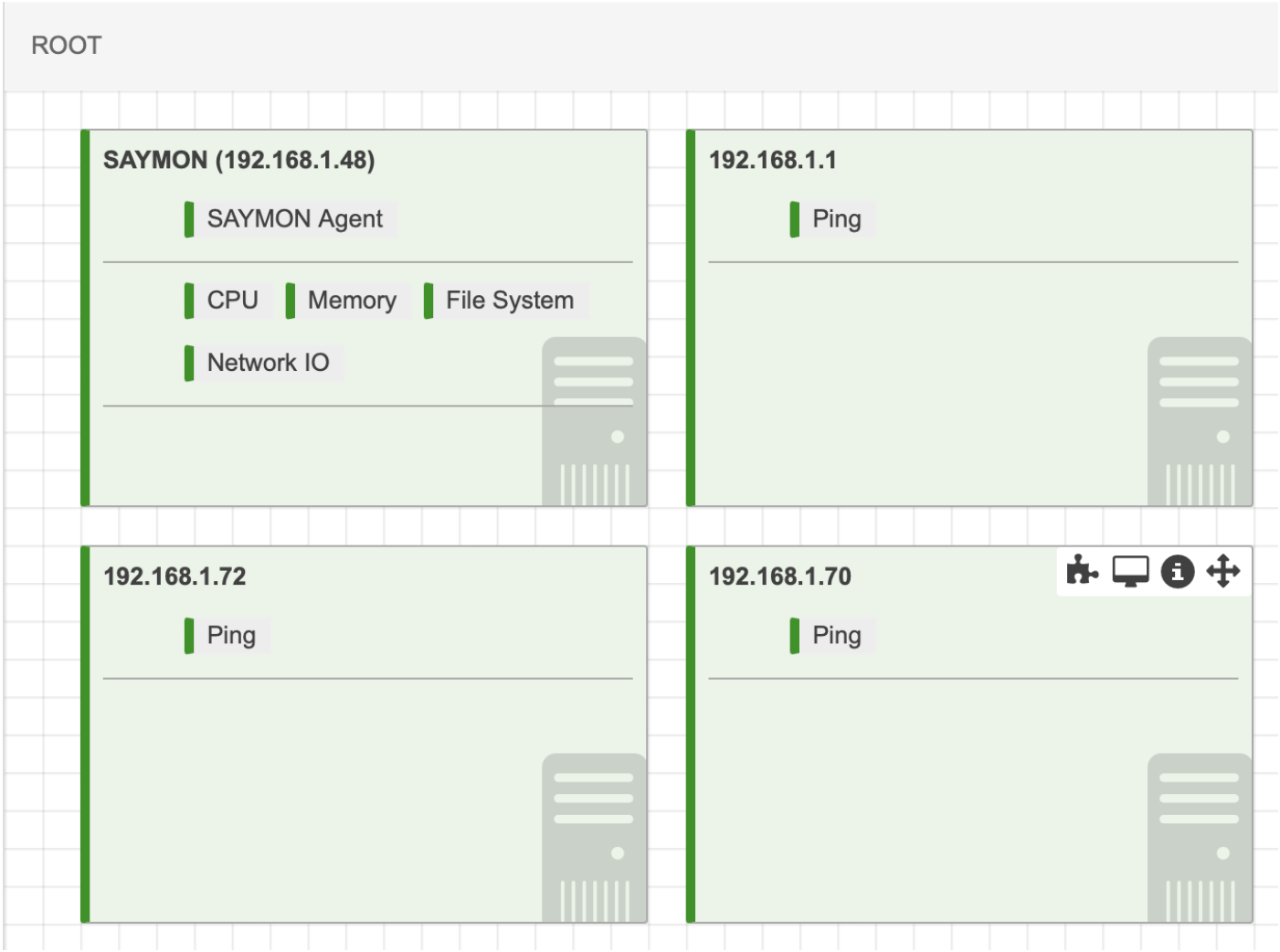
Имя сервера OpenStreetMap, отвечающего за отображения карты. Имя должно быть указано без протокола, например `openstreetmap.org` или `192.168.1.10/maps`.

По умолчанию – openstreetmap.org

Также можно использовать локальный OSM-сервер.

grid

В разделе задаются настройки сетки для выравнивания объектов в стандартном виде.



YAML

```yaml
grid:
  dim: 20
  color: rgba(128, 128, 128, 0.3)
  border: 4
```

JSON

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

grid.dim

Размер сетки (в пикселях).

WARNING	Если указано некорректное значение (например, 0 или отрицательное число), то сетка не будет отображаться, но положение объектов будет привязываться к стандартному размеру сетки (20 пикселей).

По умолчанию – **20**.

grid.**color**

Цвет сетки в формате RGBA.

По умолчанию – **rgba(128, 128, 128, 0.3)**.

grid.**border**

Максимальное значение отступа границы объекта от границы сетки (в пикселях).

Поведение объекта при изменении его размера со стандартными настройками (`border: 20`, `dim: 4`):

[grid resize border] | *web-interface-configuration/grid-resize-border.gif*

По умолчанию – **4**.

gridView

В разделе задаются параметры отображения объектов в виде сетки.

YAML

```yaml
gridView:
  cols: 6
  colWidth: 240
  rowHeight: 90
  gutterSize: 5
```

JSON

```json
{
  "gridView": {
    "cols": 6,
    "colWidth": 240,
    "rowHeight": 90,
    "gutterSize": 5
  }
}
```

gridView.**cols**

Количество объектов в строке сетки.

При **gridView.cols > 0** ширина объектов меняется автоматически.

При **gridView.cols = 0** число объектов в строке сетки подстраивается под ширину экрана.

По умолчанию – **6**.

gridView.**colWidth**

Ширина объектов при **gridView.cols = 0** (в пикселях).

По умолчанию – **240**.

gridView.**rowHeight**

Высота объектов (в пикселях).

По умолчанию – **90**.

gridView.**gutterSize**

Величина отступа между объектами (в пикселях).

По умолчанию – **5**.

history

В этом разделе задаются параметры, отвечающие за отображение данных на графиках, а также активируются некоторые дополнительные элементы интерфейса.

YAML

```yaml
history:
  downsampling:
    - period: 1-hours
      dsp: 60s-avg
    - period: 1-days
      dsp: 5m-avg
    - period: 1-weeks
      dsp: 1h-avg
    - dsp: 1d-avg
  modelChangeAnnotations: false
```

JSON

```
{
```

```
  "history": {
    "downsampling": [
      {
          "period": "1-hours",
          "dsp": "60s-avg"
      },
      {
          "period": "1-days",
          "dsp": "5m-avg"
      },
      {
          "period": "1-weeks",
          "dsp": "1h-avg"
      },
      {
          "dsp": "1d-avg"
      }
    ],
    "modelChangeAnnotations": false
  }
}
```

history.**downsampling**

Параметры уменьшения размера выборки данных – downsampling.

WARNING	Если в файл конфигурации добавлен раздел `downsampling`, то значения по умолчанию не используются. Если сервер не найдёт подходящие настройки даунсэмплинга, то вместо графика будет показана ошибка "Нет данных для выбранного диапазона".

history.downsampling.**period**

Период выборки данных.

Настройка периода имеет следующий формат:

```
<количество>-<hours|days|weeks|months>
```

history.downsampling.**dsp**

Настройка способа даунсэмплинга данных.

Настройка имеет следующий формат:

```
<количество><префикс-промежутка-времени>-<avg|min|max>
```

Доступные префиксы промежутков времени:

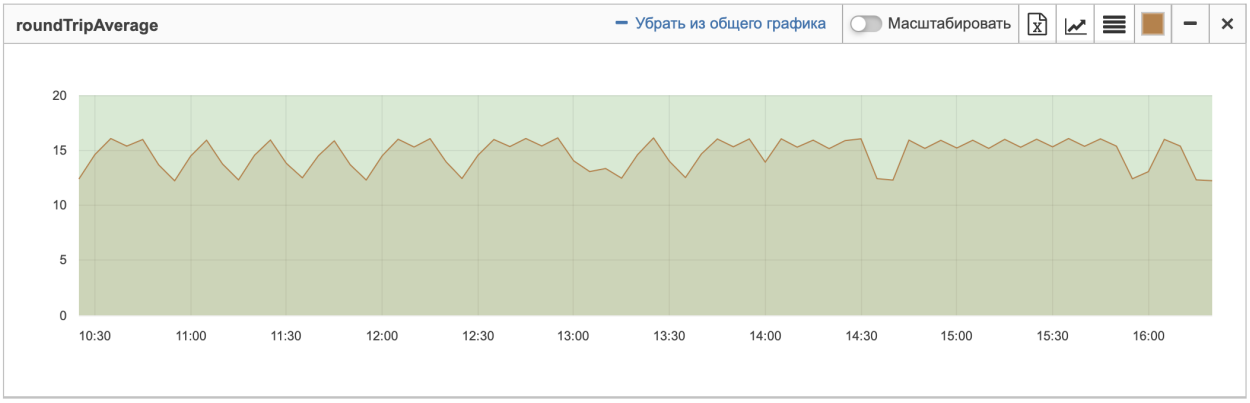
- `s` – секунды;
- `m` – минуты;
- `h` – часы;
- `d` – дни.

На примере ниже представлены примеры конфигурации способа даунсэмплинга данных.

"dsp: 5m-avg"

При следующей конфигурации, при выбранном врменном диапазоне в 6 часов, на графике будут отображены средние значения за каждые 5 минут.

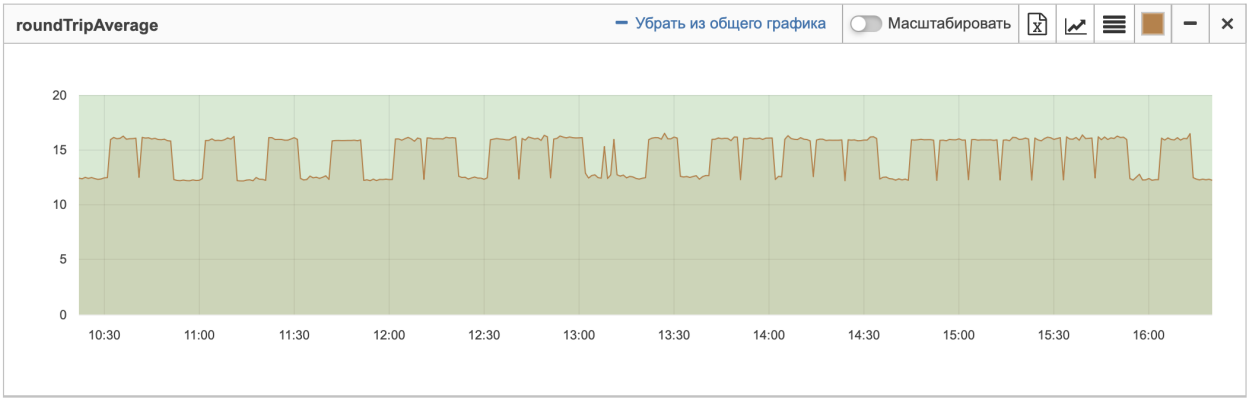
```
history:
  downsampling:
    - period: 6-hours
      dsp: 5m-avg
```



"dsp: 15s-avg"

При следующей конфигурации, при выбранном временном диапазоне в 6 часов, на графике будут отображены средние значения за каждые 15 секунд.

```
history:
  downsampling:
    - period: 6-hours
      dsp: 15s-avg
```



TIP	Если не указан параметр `period`, то такой параметр `dsp` становится параметром по умолчанию для всех периодов, которые не указаны в файле конфигурации.

history.modelChangeAnnotations

Включает смену аннотаций на графиках;

- false – смена аннотаций выключена,
- true – смена аннотаций включена.

По умолчанию – false – смена аннотаций выключена.

hideConnectionStatusNotifications

Скрывает оповещения о статусе соединения с Comet-сервером;

YAML

```yaml
hideConnectionStatusNotifications: false
```

JSON

```json
{
  "hideConnectionStatusNotifications": false,
  ...
}
```

- false – оповещения выводятся в web-интерфейс,
- true – оповещения выводятся в консоль.

По умолчанию – **false** – оповещения о статусе соединения с Comet-сервером выводятся в web-интерфейс.

hideResponseErrorNotifications

Скрывает оповещения об ошибках, возникающих при отправке автоматических запросов на сервер.

YAML

```yaml
hideResponseErrorNotifications: false
```

JSON

```json
{
  "hideResponseErrorNotifications": false,
  ...
}
```

- false – оповещения выводятся в web-интерфейс,
- true – оповещения скрыты.

По умолчанию – **false** – оповещения выводятся в web-интерфейс.

WARNING	Ошибки, возникающие при действиях пользователя, например сохранение каких-либо данных в формах, будут выводиться в web-интерфейс в любом случае.

incidents

В этом разделе настраиваются дополнительные поля для списков активных аварий и исторических аварий.

YAML

```yaml
incidents:
  additionalFields:
    - owner.properties.address
    - owner.class_id
    - updated
  additionalFieldNames:
    - IP-address
    - Object Class ID
```

JSON

```json
{
  "incidents": {
    "additionalFields": [
```

```
      "owner.properties.address",
      "owner.class_id",
      "updated"
    ],
    "additionalFieldNames": [
      "IP-address",
      "Object Class ID"
    ]
  }
}
```

incidents.**additionalFields**

Источники данных для дополнительных полей.

Источником дополнительного поля может быть любой параметр аварии. Поля параметров-объектов нужно указывать через точку. Например, чтобы указать Свойство `address` объекта, на котором произошла авария в список `additionalFields` нужно добавить значение `owner.properties.address`.

Список всех параметров можно посмотреть в документации REST API.

WARNING	Только свойства с `type_id` = 1 (свойства, добавленные через окно Свойство или через REST API) можно использовать как источники данных. Нельзя добавлять встроенные параметры и параметры сенсора, например `TaskType` или `PingHost`, а также ссылки, добавленные как Документы.

NOTE	Если источнику данных не соответствует имя поля в списке `additionalFieldNames`, то в заголовке столбца будет указано название источника данных.

incidents.**additionalFieldNames**

Имена дополнительных полей, отображаемые в интерфейсе.

Список имён соотносится со списком источников данных. То есть, первому источнику в списке `additionalFields` соответствует первое имя в списке `additionalFieldNames`, второму – второе и так далее.

Количество имён может не совпадать с количеством источников данных. Если источнику данных не соответствует имя поля, то в заголовке столбца будет указано название источника данных.

Чтобы пропустить источник данных и использовать название источника данных, нужно записать `null` или `false`.

Например, следующая конфигурация добавит в списки аварий столбцы `localTimestamp`, 'Owner ID', `IP Address` и `owner.class_id`:

```
incidents:
  additionalFields:
    - localTimestamp
    - owner._id
    - owner.properties.address
    - timestamp
    - owner.class_id
  additionalFieldNames:
    - false
    - IP-address
    - Object Class ID
```

languages

Раздел позволяет управлять предустановленными языками интерфейса web-приложения.

С помощью файла конфигурации нельзя скрыть язык из веб-интерфейса. Если не указать язык в файле конфигурации, то для него будут использованы стандартные настройки.

YAML

```
languages:
  en:
    short: us
    long: English
  ru:
    short: ru
    long: Русский
  it:
    short: it
    long: Italiano
```

JSON

```
{
  "languages": {
    "en": {
      "short": "us",
      "long": "English"
    },
    "ru": {
      "short": "ru",
      "long": "Русский"
    },
    "it": {
      "short": "it",
```

```
        "long": "Italiano"
      }
    }
  }
```

languages.**<id>**

Идентификатор языка.

Доступны три языка:

- en – английский,
- it – итальянский,
- ru – русский.

NOTE	Если указать другой язык, то он добавится в список языков в меню текущего пользователя, но будет использовать английскую локализацию.

languages.<id>.**long**

Полное название языка.

Это название отображается в меню текущего пользователя Центрального Пульта при выборе языка интерфейса.

languages.<id>.**short**

Идентификатор иконки флага.

Иконка отображается в меню текущего пользователя Центрального Пульта при выборе языка интерфейса.

 – es,

 – gb,

 – it,

 – ru,

 – us.

NOTE	Если указать несуществующий или некорректный идентификатор, то будет использован флаг .

monitoring

В этом разделе настраивается скрытие стандартных мониторинговых сенсоров из интерфейса.

Если этот раздел не задан в файле конфигурации, то в интерфейсе отображаются все стандартные сенсоры.

YAML

```yaml
monitoring:
  standardTasks:
    - ping
    - snmpGet
```

JSON

```json
{
  "monitoring": {
    "standardTasks": [
      "ping",
      "snmpGet"
    ]
  }
}
```

monitoring.**standardTasks**

Список идентификаторов отображаемых стандартных сенсоров. Список идентификаторов можно посмотреть в разделе Список идентификаторов стандартных сенсоров.

millisecondMonitoring

Включает обновление данных каждую миллисекунду.

YAML

```yaml
millisecondMonitoring: false
```

JSON

```json
{
  "millisecondMonitoring": false,
  ...
}
```

- false – обновление выключено,
- true – обновление включено.

По умолчанию – **false** – обновление выключено.

pollInterval

Интервал между запросами данных с REST-сервера, если соединение между клиентом и Comet-сервером недоступно (в миллисекундах).

YAML

```
pollInterval: 5000
```

JSON

```
{
  "pollInterval": 5000,
  ...
}
```

По умолчанию – **5000** – 5 секунд.

pollIntervalSocket

Интервал между запросами данных с Comet-сервера, если соединение между клиентом и Comet-сервером доступно (в миллисекундах).

YAML

```
pollIntervalSocket: 60000
```

JSON

```
{
  "pollIntervalSocket": 60000,
  ...
}
```

По умолчанию – **60000** – 1 минута.

statHistory

Отображать кнопку "Показать историю данных".

YAML

```yaml
statHistory: false
```

JSON

```
{
  "statHistory": false,
  ...
}
```

- true – кнопка включена,
- false – кнопка выключена.

NOTE	Параметр серверной конфигурации agent_stat должен быть включён.

По умолчанию – **false**

title

Заголовок для вкладок, а также имя, отображаемое в левом верхнем углу web-интерфейса.

YAML

```yaml
title: "SAYMON"
```

JSON

```
{
  "title": "SAYMON",
  ...
}
```

По умолчанию – **SAYMON**.

themes

Этот параметр отвечает за настройку тем.

YAML

```yaml
themes:
  default:
    name: Light
```

```yaml
  dark:
    name: Dark
```

JSON

```json
{
  "themes": {
    "default": {
        "name": "Light"
    },
    "dark": {
        "name": "Dark"
    }
  }
}
```

CSS-файлы с темами находятся в директории `/usr/local/saymon/client/css`.

По умолчанию доступно две темы – светлая (`default`) и тёмная (`dark`).

Чтобы добавить новую тему, нужно создать в директории с темами новую папку (например, `my-custom-theme`). Затем, добавить эту папку в файл конфигурации:

YAML

```yaml
themes:
  my-custom-theme:
    name: Custom Theme
```

JSON

```json
{
    "themes": {
        "my-custom-theme": {
            "name": "Custom Theme"
        }
    }
}
```

themes.<theme-name>.**name**

Название темы в меню текущего пользователя.

NOTE	Название темы нельзя локализовать на разные языки.

tree

В этом разделе задаются настройки для дерева объектов.

YAML

```yaml
tree:
  filterSubmit: true
```

JSON

```json
{
    "tree": {
        "filterSubmit": true
    },
}
```

tree.filterSubmit

Включает кнопку применения фильтра и отключает его автоматическое применение при вводе символов.

Кнопка применения фильтра отображается рядом с кнопкой сброса фильтра при вводе нового фильтра:



- false – кнопка выключена,
- true – кнопка включена.

По умолчанию – **false** – кнопка выключена.

useNaturalSort

Включает сортировку объектов с помощью библиотеки javascript-natural-sort:

YAML

```yaml
useNaturalSort: false
```

JSON

```
{
  "useNaturalSort": false,
  ...
```

```
}
```

- false – объекты сортируются с помощью нативных функций браузера,
- true – объекты сортируются с помощью библиотеки javascript-natural-sort.

По умолчанию – **false** – объекты сортируются с помощью нативных функций браузера.

CAUTION	При установленном useNaturalSort = true возможно замедление работы web-интерфейса при большом количестве объектов в системе (от 100 000).