Руководство администратора

Центральный Пульт 3.18.93

Содержание

1. Установка сервера 1
1.1. Установка сервера 1
1.1.1. Требования для установки сервера системы 1
1.2. Установка системы Центральный Пульт на виртуальную машину Oracle VM VirtualBox 2
1.2.1. Подготовка виртуальной машины 2
1.2.2. Установка системы 5
1.2.3. Вход в систему 7
2. Конфигурация 9

2.1. Конфигурация системы 9
2.1.1. Архитектура системы 10
2.1.1.1. Описание взаимодействия 10
2.1.1.2. Отказоустойчивость и резервирование 12
2.1.1.3. Версии используемого и поддерживаемого ПО 13
2.1.2. Сетевые доступы 14
2.1.3. Файлы конфигурации 16
2.1.3.1. Конфигурация сервера 17
2.1.3.2. Конфигурация Web-приложения 50
2.1.3.3. Конфигурация журналирования сервера 88
2.2. Инструкции 95
2.2.1. Настройка отправки SMS-уведомлений и голосовых вызовов 95
2.2.2. Настройка пользовательских сенсоров мониторинга 97
2.2.2.1. Каталог для пользовательских сенсоров 98
2.2.2.2. Параметры пользовательского сенсора 99
2.2.2.3. Пример пользовательского сенсора 102
2.2.2.4. Использование утилит операционной системы 104
2.2.3. Скрытие неиспользуемых сенсоров мониторинга 106
2.2.3.1. Список идентификаторов стандартных сенсоров 106
2.2.4. Выполнение скриптов при создании и удалении объектов 108
2.2.4.1. Каталог для скриптов 108
2.2.4.2. Параметры скриптов 109
2.2.4.3. Пример скрипта 111
2.2.5. Модуль аналитики 115
2.2.5.1. Установка модуля аналитики 115
2.2.5.2. Включение модуля аналитики 115
2.2.5.3. Использование модуля аналитики 116
2.2.5.4. Мониторинг работы модуля аналитики 116
2.2.5.5. Производительность модуля аналитики 116
2.2.6. Подключение и настройка Elasticsearch 118
2.2.6.1. Настройка и запуск ELK 118
2.2.6.2. Настройка системы Центральный Пульт для работы с ELK 119
2.2.6.3. Получение данных из Elasticsearch 120
2.2.7. Передача аварийных сообщений в сторонние сервисы 121
2.2.8. Самоконтроль сервера 122
2.2.8.1. Использование памяти сервером 123
2.2.8.2. Статистики использования памяти 124
2.2.9. Запуск сервера через PM2 125
2.2.9.1. Установка PM2 125
2.2.9.2. Конфигурация акторов 125
2.2.9.3. Отключение сервиса 125
2.2.9.4. Запуск через PM2 125
2.2.9.5. Настройка журнала 126

2.2.9.6. Работа с акторами 126
2.2.10. Включение самостоятельной регистрации пользователей 128
2.2.11. Keycloak 129
2.2.11.1. Учётные записи Keycloak-пользователей 129
2.2.11.2. Удаление Keycloak-пользователей 129
2.2.12. LDAP 130
2.2.12.1. Настройка LDAP 130
2.2.12.2. Аутентификация в системе Центральный Пульт через LDAP 130
3. Подключение к серверу 133
3.1. Подключение к системе через Web-интерфейс 133
3.2. Подключение к системе через REST API 134
3.3. Подключение к серверу по SSH 135
4. Агент SAYMON 136

4.1. Установка агента ... 137
4.1.1. Требования для установки агента системы ... 137
4.1.2. Windows ... 138
4.1.2.1. Установка ... 138
4.1.2.2. Log-файлы ... 141
4.1.2.3. Поддерживаемые версии ОС ... 141
4.1.3. Linux ... 142
4.1.3.1. Установка с помощью скрипта ... 142
4.1.3.2. Установка с помощью пакетного менеджера ... 144
4.1.3.3. Регистрация агента ... 147
4.1.3.4. Журналирование ... 147
4.1.3.5. Работа службы агента ... 147
4.1.3.6. Поддерживаемые версии Linux ... 147
4.1.4. macOS ... 148
4.1.4.1. Установка ... 148
4.1.4.2. Место установки ... 148
4.1.4.3. Log-файлы ... 148
4.1.5. Wiren Board ... 149
4.1.5.1. Установка ... 149
4.1.5.2. Конфигурация агента ... 149
4.1.5.3. Место установки ... 149
4.1.5.4. Файлы журнала ... 149
4.1.5.5. Работа службы ... 150
4.1.6. Автоинсталляция ... 151
4.1.7. Установка дополнительных агентов ... 153
4.1.7.1. Linux ... 153
4.1.7.2. Windows ... 154
4.1.8. Конфигурационный файл агента ... 155
4.1.8.1. Подключение к Kafka Cluster ... 158
4.1.8.2. Авторизация и шифрование ... 159
4.1.9. Конфигурация log-файлов агента ... 160
4.1.9.1. Базовое журналирование ... 160
4.1.9.2. Активация debug-режима ... 160
4.1.10. Автообнаружение ... 162
4.1.10.1. Встроенное автообнаружение ... 162
4.1.10.2. Автообнаружение через API ... 164
4.1.11. Обновление агентов ... 166
4.1.11.1. Windows ... 166
4.1.11.2. macOS ... 166
4.1.11.3. Linux ... 167
4.1.11.4. Обновление через web-интерфейс ... 170
4.1.12. Метаданные агента ... 173
4.1.12.1. Поля метаданных ... 173

5. Мониторинг 176

5.1. Архитектура мониторинга в системе Центральный Пульт ... 176
5.2. Выбор агента ... 178
5.3. Встроенные сенсоры ... 179
5.3.1. Бинарный протокол ... 180
5.3.1.1. Пример настройки ... 180
5.3.1.2. Результаты выполнения ... 180
5.3.1.3. Известные ошибки при настройке сенсора ... 181
5.3.2. Конфигурационный файл / директория ... 182
5.3.2.1. Пример настройки ... 182
5.3.2.2. Результат выполнения ... 182
5.3.2.3. Известные ошибки при настройке сенсора ... 183
5.3.3. Запрос в базу данных ... 184
5.3.3.1. Пример настройки ... 184
5.3.3.2. Результаты выполнения ... 185
5.3.3.3. Консольная программа ... 185
5.3.3.4. Известные ошибки при настройке сенсора ... 186
5.3.4. FTP-сенсор ... 188
5.3.4.1. Пример настройки ... 188
5.3.4.2. Результаты выполнения ... 189
5.3.4.3. Консольная программа ... 190
5.3.4.4. Известные ошибки при настройке сенсора ... 190
5.3.5. HTTP-запрос ... 192
5.3.5.1. Пример настройки ... 193
5.3.5.2. Результат выполнения ... 194
5.3.5.3. Консольная программа ... 194
5.3.5.4. Известные ошибки при настройке сенсора ... 196
5.3.6. JMX-сенсор ... 197
5.3.6.1. Пример настройки ... 197
5.3.6.2. Результаты выполнения ... 198
5.3.6.3. Известные ошибки при настройке сенсора ... 198
5.3.7. Локальный порт ... 199
5.3.7.1. Пример настройки ... 199
5.3.7.2. Результат выполнения ... 199
5.3.7.3. Консольная программа ... 200
5.3.7.4. Известные ошибки при настройке сенсора ... 200
5.3.8. MQTT-сенсор ... 201
5.3.8.1. Пример настройки ... 201
5.3.8.2. Результат выполнения ... 201
5.3.8.3. Консольная программа ... 201
5.3.9. Пинг-сенсор ... 203
5.3.9.1. Пример настройки ... 203
5.3.9.2. Результат выполнения ... 203
5.3.9.3. Консольная программа ... 204

5.3.9.4. Известные ошибки при настройке сенсора 205
5.3.10. Процесс по имени 207
5.3.10.1. Пример настройки 207
5.3.10.2. Результат выполнения 207
5.3.10.3. Консольная программа 209
5.3.11. Выполнение программы/скрипта 211
5.3.11.1. Настройка 211
5.3.11.2. Редактор скриптов 212
5.3.11.3. Примеры 213
5.3.12. Удалённый порт 217
5.3.12.1. Пример настройки 217
5.3.12.2. Результат выполнения 217
5.3.12.3. Консольная программа 218
5.3.12.4. Известные ошибки при настройке сенсора 218
5.3.13. SNMP Get 219
5.3.13.1. Пример настройки 219
5.3.13.2. Результат выполнения 221
5.3.13.3. Консольная программа 222
5.3.14. SNMP Trap 223
5.3.14.1. Пример настройки 223
5.3.14.2. Результат выполнения 224
5.3.14.3. Консольная программа 225
5.3.14.4. Использование SNMP v3 225
5.3.15. WMI-сенсор 226
5.3.15.1. Пример настройки 226
5.3.15.2. Результат выполнения 226
5.3.15.3. Консольная программа 227
5.3.16. Agent Watchdog 228
6. Безопасность 230

6.1. Рекомендации по обеспечению информационной безопасности ... 230
6.1.1. Смена пароля по умолчанию ... 230
6.1.2. Ограничение доступа к базам данных ... 230
6.1.3. Ограничение доступа к конфигурационным файлам ... 230
6.1.4. Использование сертификата безопасности SSL ... 231
6.1.4.1. Самоподписанные сертификаты ... 231
6.1.4.2. Бесплатные сертификаты ... 231
6.1.4.3. Коммерческие сертификаты ... 232
6.1.4.4. Регистрация доменного имени для web-сервера ... 233
6.1.4.5. Получение бесплатного SSL-сертификата от Let's Encrypt ... 234
6.1.4.6. Создание самоподписанного SSL-сертификата ... 238
6.1.4.7. Получение коммерческого сертификата ... 246
6.1.5. Шифрование паролей ... 247
6.1.5.1. Формат зашифрованных параметров ... 247
6.1.5.2. Шифрование файлов конфигурации ... 247
6.1.5.3. Шифрование параметров проверок ... 248
6.1.6. Настройка Comet- и Rest-сервера для использования сертификата безопасности SSL ... 249
6.1.6.1. Настройка Comet-сервера ... 249
6.1.6.2. Настройка Rest-сервера ... 249
6.1.7. Настройка Kafka для использования сертификата безопасности SSL ... 250
6.1.8. Настройка защищенного подключения через протокол MQTT ... 251
6.1.9. Журнал событий безопасности ... 252
6.1.9.1. Архитектура журналирования событий безопасности ... 252
6.1.9.2. Включение журнала безопасности ... 254
6.1.9.3. События журнала безопасности ... 256
6.1.9.4. Поддержка формата CEF ... 258
7. Расширения ... 261
7.1. Расширения ... 261
7.1.1. Установка расширений ... 261
7.1.2. Пользовательские расширения ... 261
8. Интеграции ... 264

8.1. Интеграция с Zabbix ... 265
8.1.1. Настройка подключения к Zabbix 5.x ... 265
8.1.1.1. Шаг 1: проверить доступ от сервера системы до сервера Zabbix ... 265
8.1.1.2. Шаг 2: добавить пользователя в Zabbix ... 265
8.1.1.3. Шаг 3: создать объект для импорта данных ... 265
8.1.1.4. Шаг 4: включить импорт данных на сервере системы ... 265
8.1.1.5. Шаг 5: применить настройки ... 266
8.1.2. Настройка подключения к Zabbix 6.0 ... 266
8.1.2.1. Шаг 1: запуск Zabbix ... 266
8.1.2.2. Шаг 2: настройка автоматического запуска Zabbix ... 267
8.1.2.3. Шаг 3: загрузка и запуск Zabbix Agent ... 267
8.1.2.4. Шаг 4: подключение Zabbix Agent ... 268
8.1.2.5. Шаг 5: создание объекта для мониторинга ... 270
8.1.2.6. Шаг 6: создание триггера ... 271
8.1.2.7. Шаг 7: настройка подключения к серверу "Центральный Пульт" ... 272
8.1.2.8. Шаг 8: управление событием ... 273
8.1.3. Периодичность обновления данных ... 273
8.1.4. Изменение стандартных опций в запросах Zabbix ... 274
8.1.5. Корреляция класса создаваемых объектов с их именами ... 275
8.1.6. Журналирование ... 275
8.2. Интеграция с Grafana ... 276
8.3. Интеграция с Keycloak ... 281
8.3.1. Настройка сервера ... 281
8.3.1.1. Realm Certificate ... 281
8.3.1.2. Backend-клиент ... 283
8.3.1.3. Конфигурация сервера Центральный Пульт ... 286
8.3.2. Настройка web-интерфейса ... 287
8.3.2.1. Frontend-клиент ... 287
8.3.2.2. Конфигурация web-интерфейса системы Центральный Пульт ... 291
8.4. Интеграция с VictoriaMetrics ... 292
8.4.1. Шаг 1: Установка VictoriaMetrics ... 292
8.4.2. Шаг 2: Настройка конфигурации сервера ... 292
8.5. Интеграция с InfluxDB ... 293
8.5.1. Шаг 1: Установка InfluxDB ... 293
8.5.2. Шаг 2: Настройка конфигурации сервера ... 293

1. Установка сервера

1.1. Установка сервера

Серверная часть системы может быть поставлена в виде готового образа, который можно установить в виртуальную машину.

Ссылку на скачивание образа виртуальной машины с демо-версией системы можно получить на электронную почту на странице Скачать SAYMON нашего сайта.

WARNING

По умолчанию агент инициализирует подключение к серверу по порту `9092`. Необходимо проверить доступность данного порта с виртуальных машин, серверов и прочих устройств, на которых будет производиться установка агента.

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
```

1.1.1. Требования для установки сервера системы

Система может быть установлена на выделенных аппаратных или виртуальных мощностях.

Минимальные аппаратные требования для работы сервера системы:

- Операционная система:
  - Ubuntu Server 20.04;
  - Ubuntu Server 18.04;

  NOTE

  Установка системы также возможна на РЕД ОС, Red Hat Enterprise Linux, Astra Linux, Debian.
- Процессор – 4 ядра, архитектура x86_64;
- Оперативная память – 8 Гб;
- Жёсткий диск – 80 Гб.

1.2. Установка системы Центральный Пульт на виртуальную машину Oracle VM VirtualBox

Система Центральный Пульт распространяется как образ установочного диска Ubuntu Server с предустановленным сервером системы Центральный Пульт в формате ISO. Скачать образ диска можно на https://saymon.info/download/.

Образ установочного диска с системой можно использовать для установки сервера как на физическое устройство, так и на виртуальную машину. В этой статье описана установка сервера Центральный Пульт на виртуальную машину Oracle VirtualBox. При установке системы на физическое устройство начните с раздела Установка системы.

1.2.1. Подготовка виртуальной машины

Для корректной работы системы, виртуальная машина должна иметь следующие настройки:

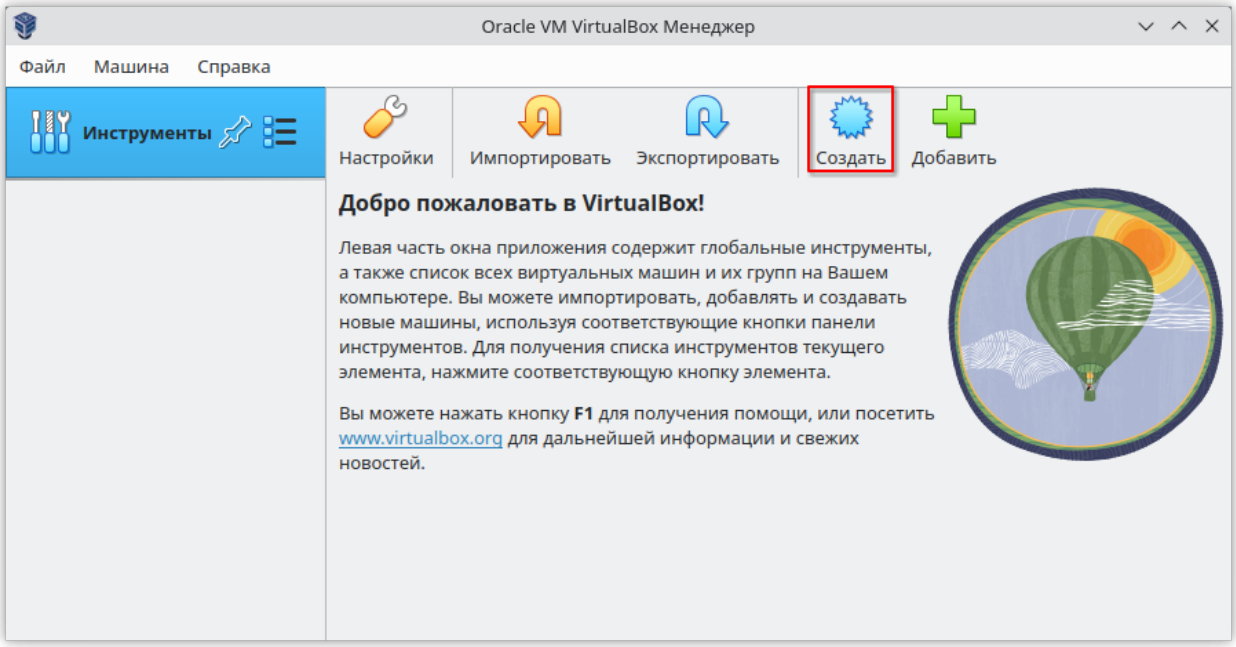
- **Оперативная память** – от 4096 МБ;
- **Процессоры** – от 2 (рекомендуется от 4);
- **Размер виртуального жёсткого диска** – от 40 ГБ (рекомендуется от 80 ГБ);
- **Версия ОС** – Ubuntu (64-bit);
- **Тип подключения сетевого адаптера** – Сетевой мост.

WARNING

Также, при установке должна быть включена опция **Пропустить автоматическую установку**.

Для создания новой виртуальной машины с этими настройками, нужно выполнить следующие шаги:

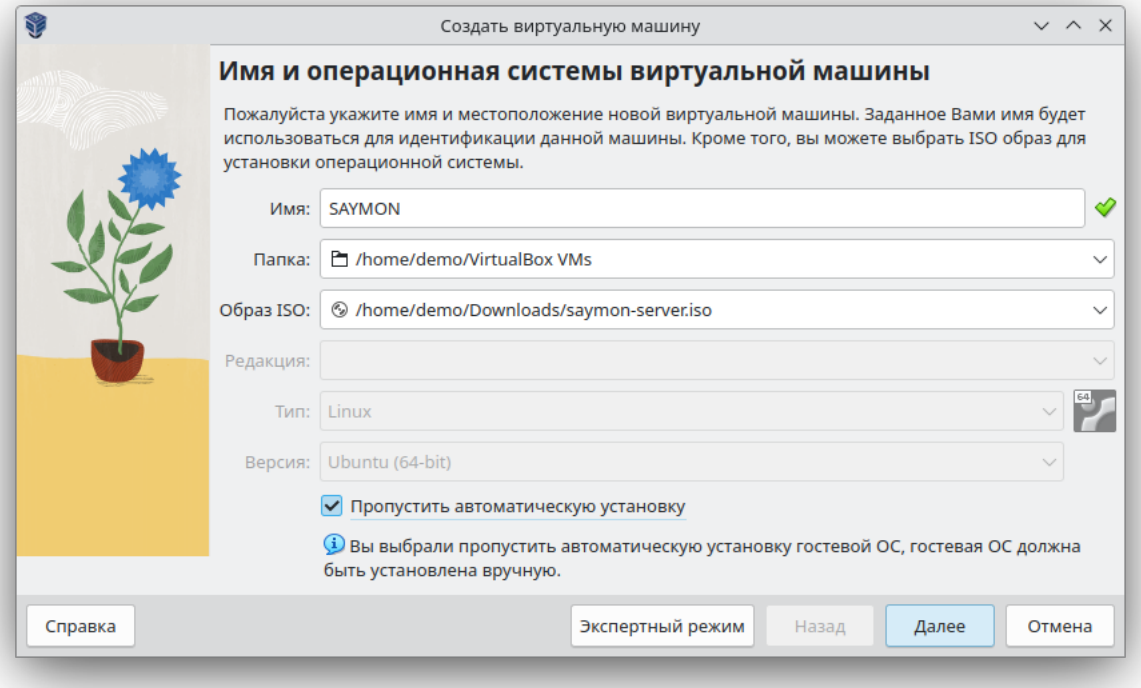
1. Нажать кнопку **Создать**.



2. Выбрать название виртуальной машины и установочный образ сервера.

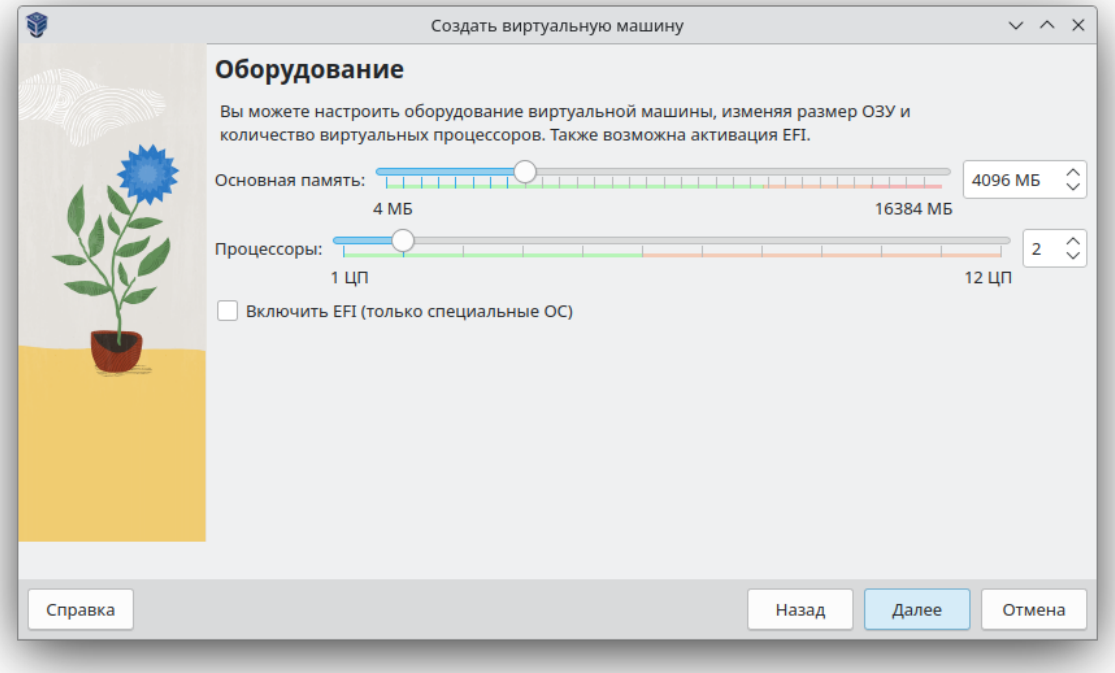
WARNING

Для корректной установки сервера, опция **Пропустить автоматическую установку** должна быть включена.

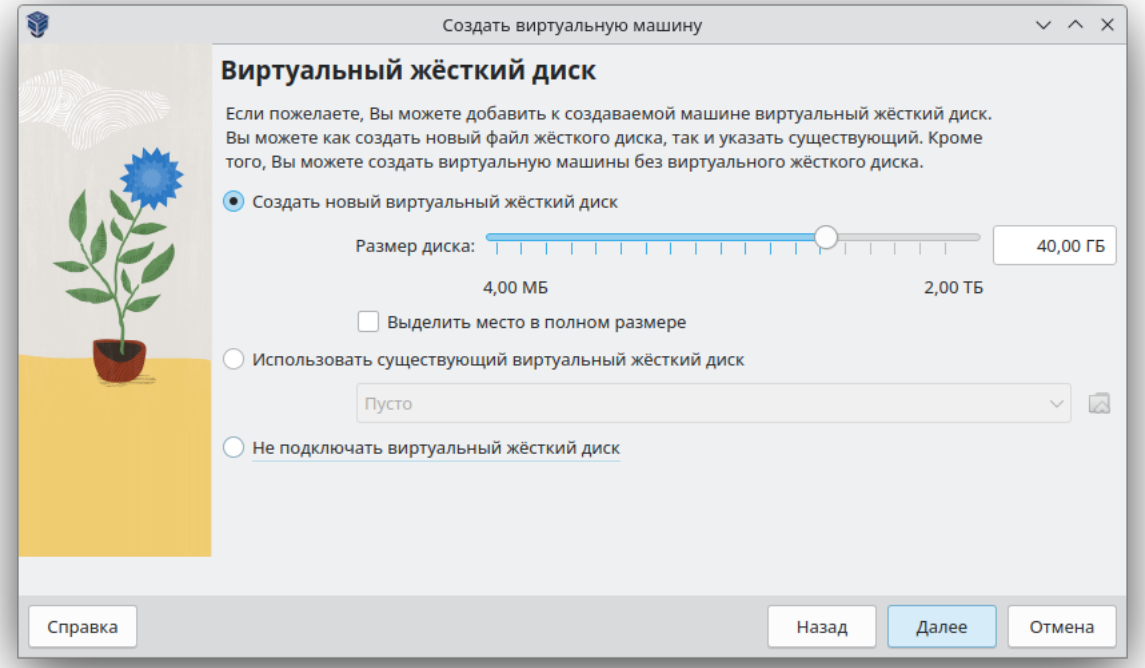


3. Настроить оборудование виртуальной машины следующим образом:

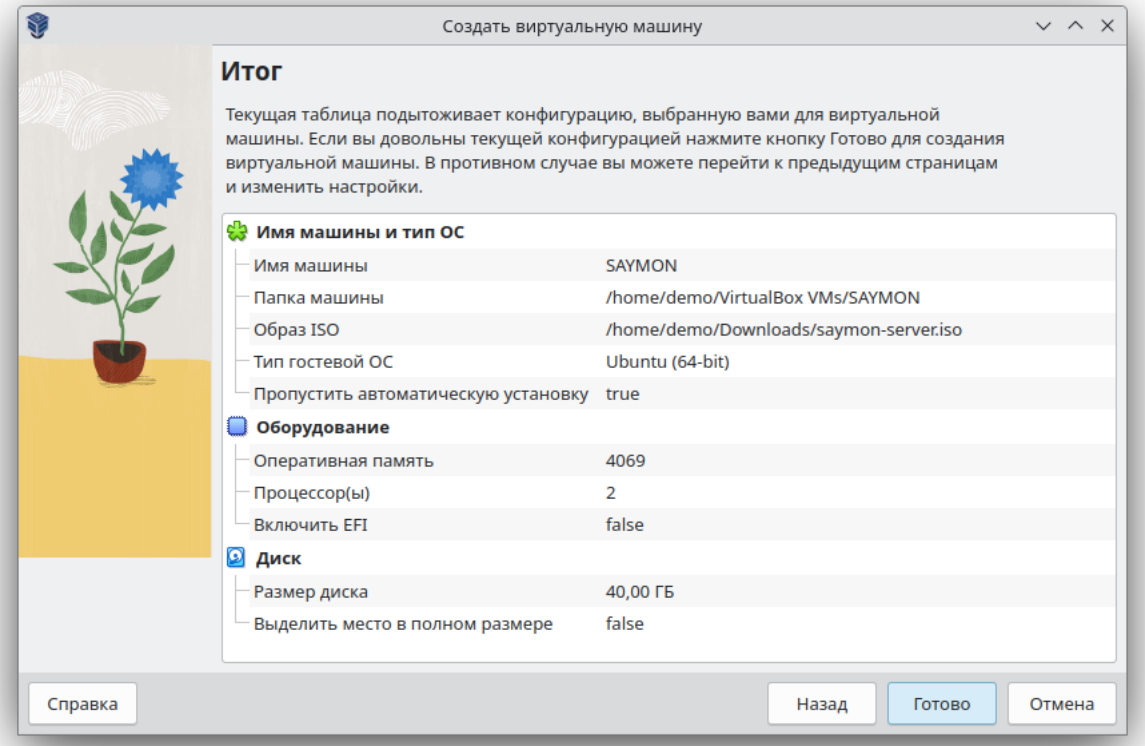
   - **Оперативная память** – не менее 4096 МБ;
   - **Процессоры** – не менее 2 (рекомендуется от 4).



4. Создать виртуальный жёсткий диск или использовать существующий. Размер жёсткого диска должен быть не менее 40 ГБ:

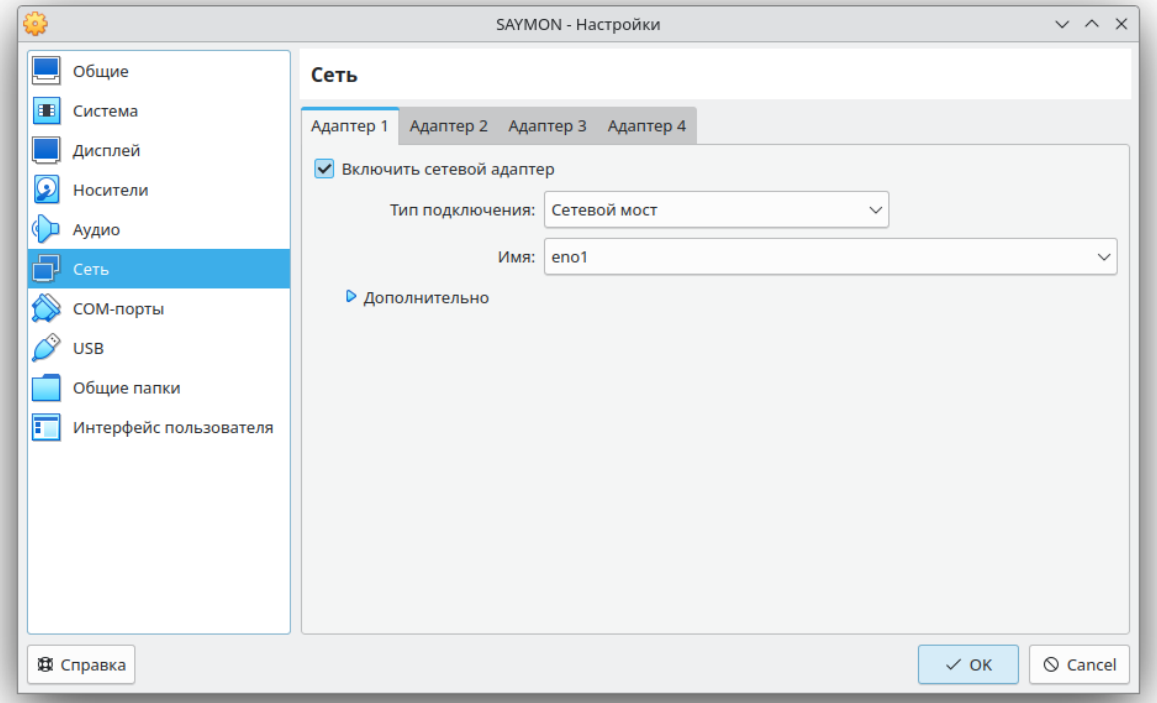


5. После создания виртуальной машины, настройки должны выглядеть следующим образом:



6.

Зайти в настройки созданной виртуальной машины. В разделе "Сеть" установить **Тип подключения** – *Сетевой Мост*.

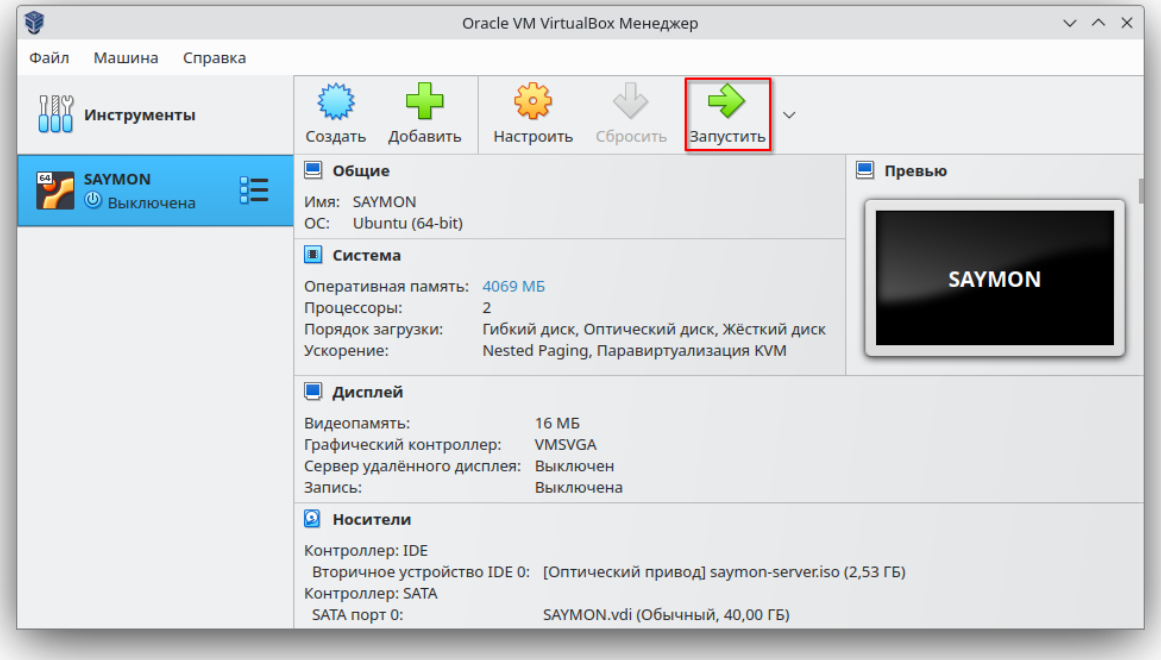


Созданную виртуальную машину можно использовать для установки системы Центральный Пульт.

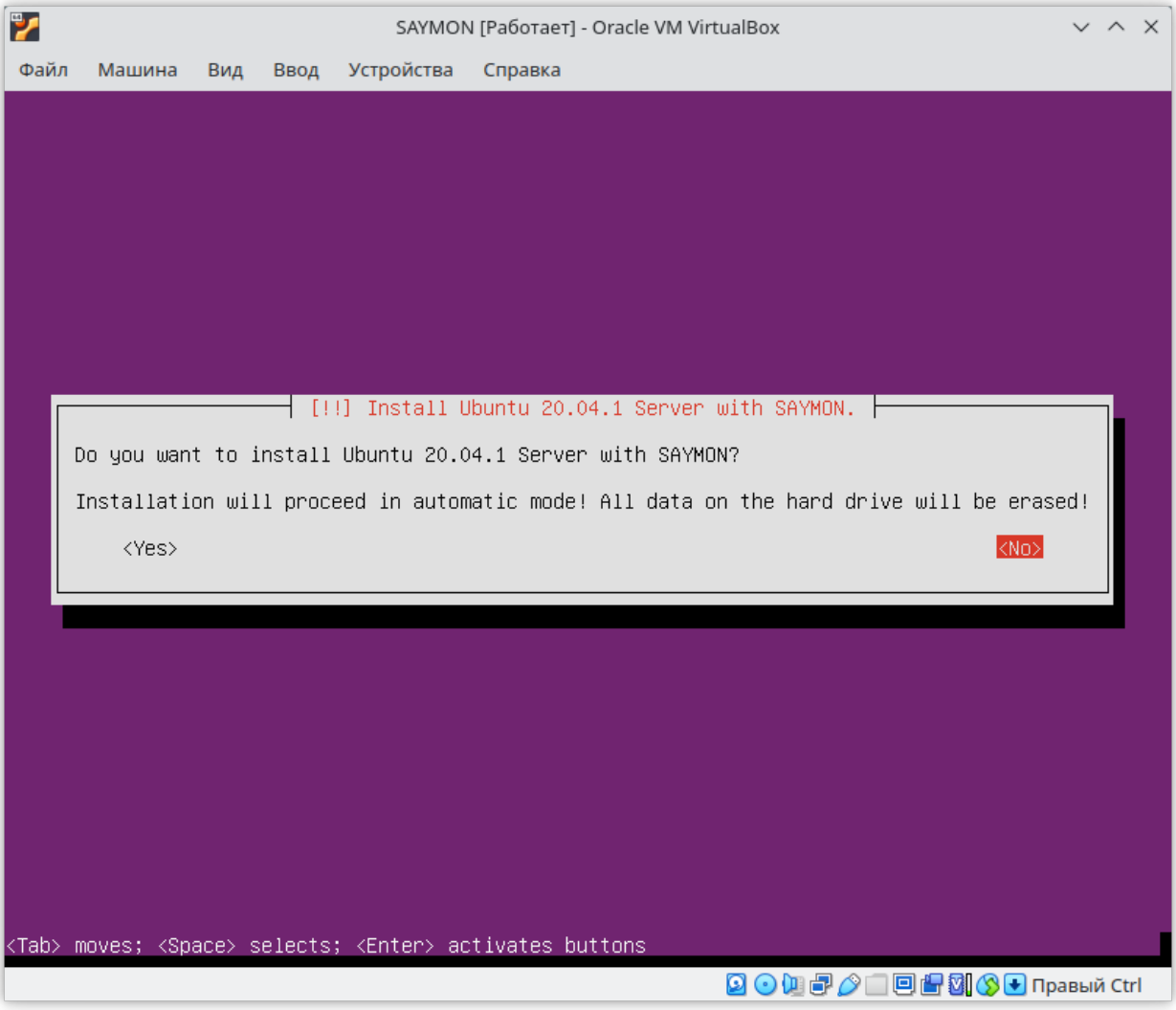
1.2.2. Установка системы

Установка системы происходит в автоматическом режиме (англ.). От пользователя требуется только подтвердить начало установки, а также перезапустить виртуальную машину после успешной установки.

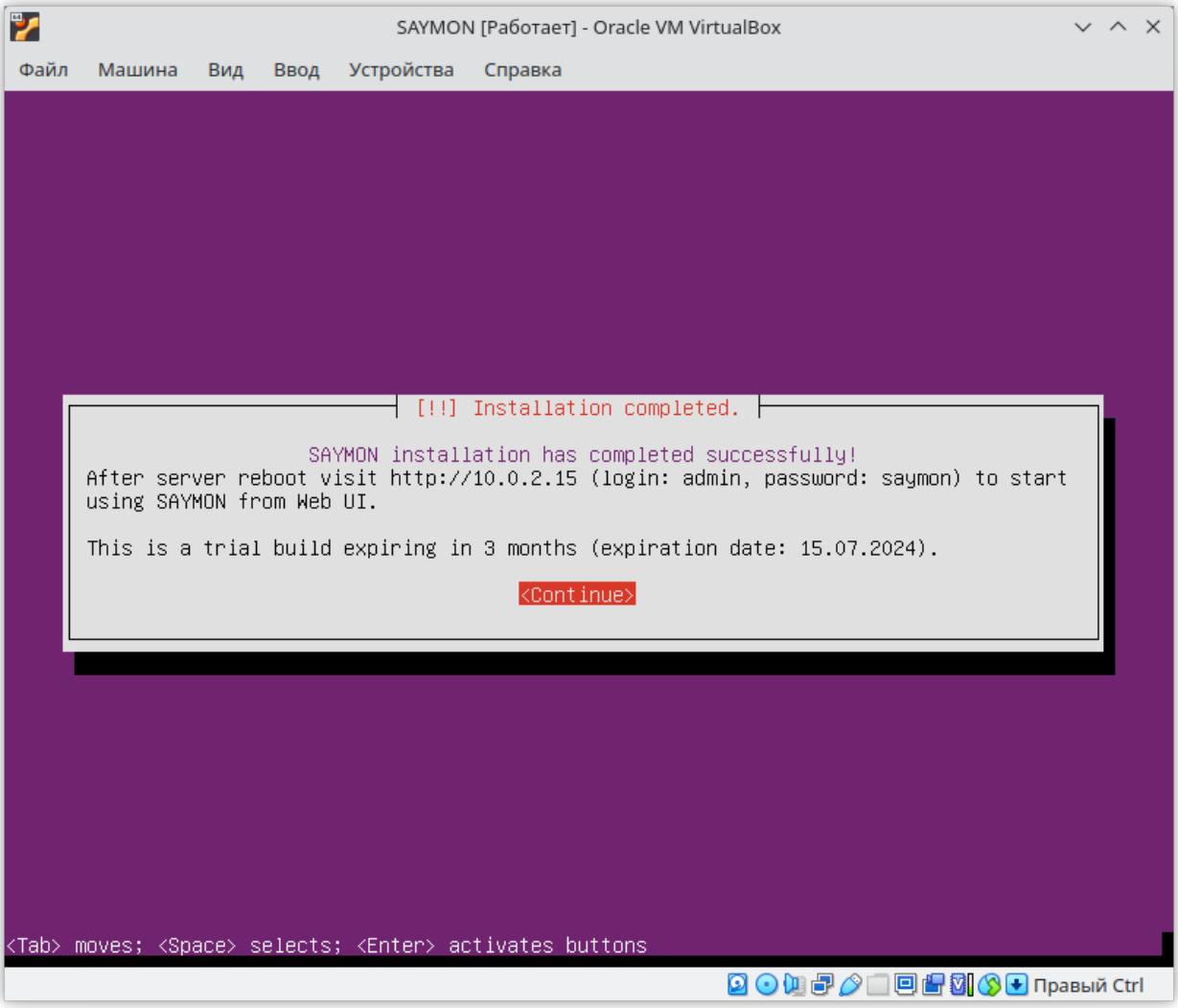
1. Для начала установки, выбрать созданную виртуальную машину и нажать кнопку **Запустить**.



2. В открывшемся окне выбрать **<Yes>** и нажать клавишу `Enter`. Начнётся процесс автоматической установки Ubuntu Server с серверной частью системы Центральный Пульт.



3. После завершения установки отобразится диалоговое окно, в котором будет указан адрес для доступа к web-интерфейсу серверной части системы Центральный Пульт (IP-адрес виртуальной машины) и данные для входа (по умолчанию **login: admin, password: saymon**). Нажмите кнопку `Enter` для закрытия окна и перезагрузки виртуальной машины.



NOTE

Пара логин-пароль для администратора сервера и для администратора системы Центральный Пульт отличаются.

1.2.3. Вход в систему

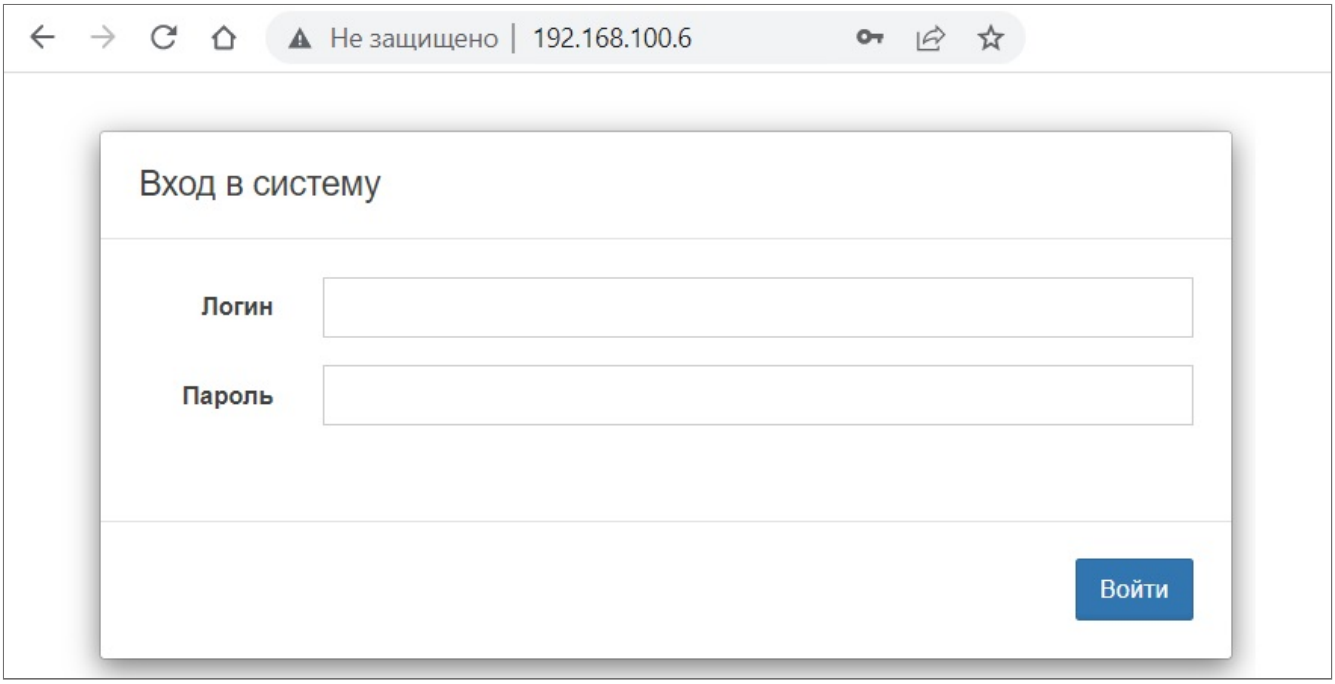
Для входа в Web-интерфейс системы Центральный Пульт необходимо перейти по IP адресу, указанному после установки системы и ввести логин и пароль администратора на экране входа в систему.

Стандартные данные для входа в учётную запись администратора:

- **Логин** - `admin`;
- **Пароль** - `saymon`.

NOTE

При первом входе через учётную запись `admin` система потребует смены пароля.



Для доступа к виртуальной машине по SSH необходимо использовать следующие данные:

- **Логин** - `saymon`;
- **Пароль** - `saymon`.

2. Конфигурация

2.1. Конфигурация системы

Этот раздел содержит справку файлов конфигурации сервера и Web-приложения, а также инструкции по настройке отдельных элементов системы.

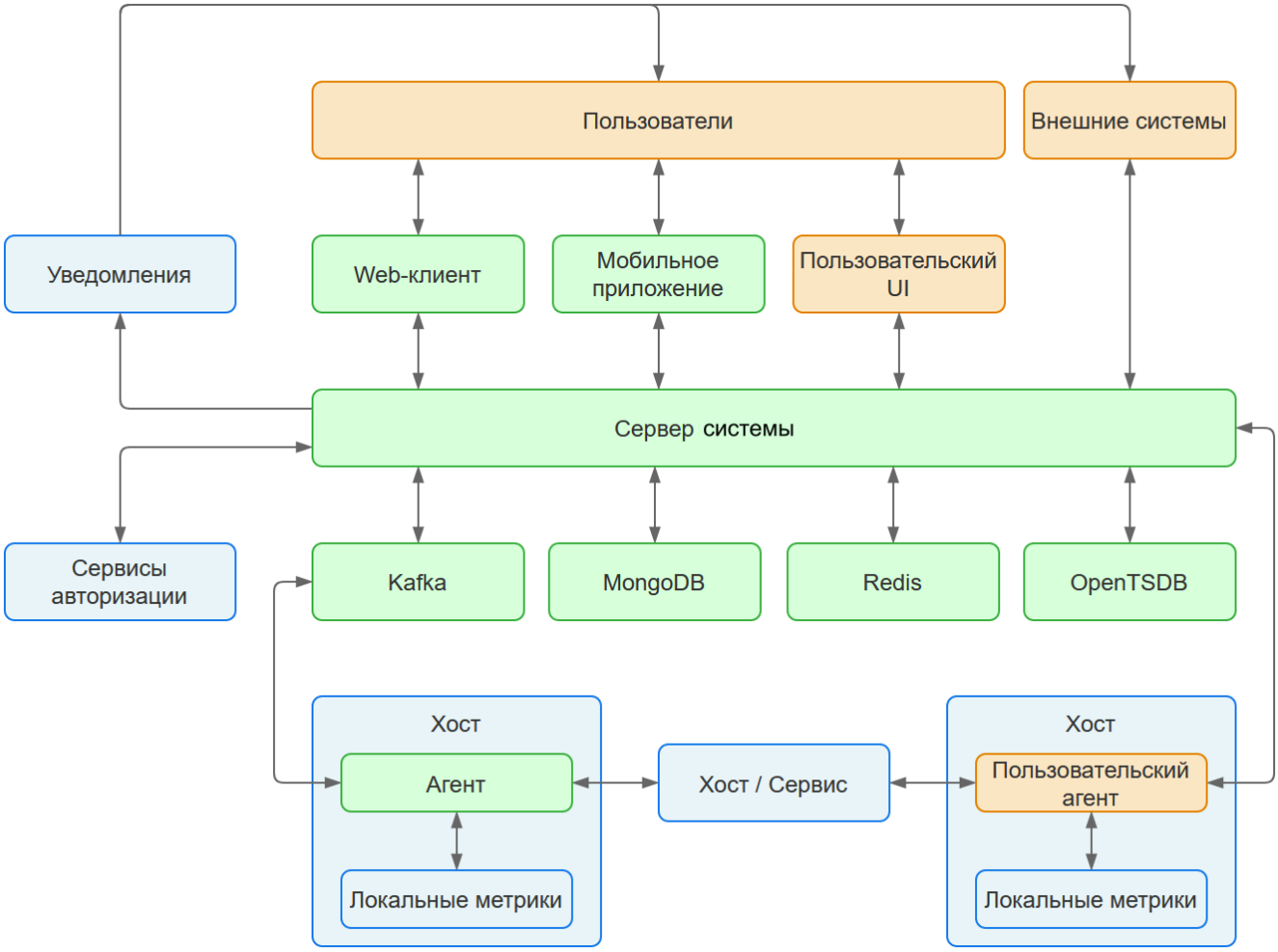
Также, в этом разделе находится описание архитектуры системы и сетевых доступов, необходимых для настройки системы.

Описание окна конфигурации системы в Web-интерфейсе находится в разделе web-interface:configuration.html.

2.1.1. Архитектура системы

2.1.1.1. Описание взаимодействия

Ниже приводится общая схема архитектуры системы Центральный Пульт.



Сервер "Центральный Пульт" – серверная часть системы. Представляет собой набор микросервисов, которые производят обработку и анализ поступивших данных, в частности, управляют логикой смены состояний у объектов и связей. В состав серверной части также входят NGINX и REST-сервер.

Kafka – шина данных, которая обеспечивает обмен данными между микросервисами системы Центральный Пульт, отправляет Агентам список задач и принимает результат их выполнения.

NOTE

Взаимодействие с устаревшими Агентами версии ниже 3.0 обеспечивается через Redis.

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

Redis – это хранилище данных в оперативной памяти (in-memory кэш), благодаря которому реализуется быстрый доступ к часто запрашиваемой информации.

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

NGINX – это HTTP-сервер, обслуживающий запросы клиентов. Отдаёт статические элементы web-интерфейса (HTML, JavaScript, CSS), а также проксирует запросы от клиента к REST-серверу.

REST-сервер – один из микросервисов системы Центральный Пульт, который отвечает за обработку REST-запросов от клиентов. Через REST API клиент получает и изменяет информацию, обрабатываемую сервером системы Центральный Пульт.

Агент – это программа, устанавливаемая на виртуальный или физический сервер или иной узел сети и осуществляющая выполнение проверок, сбор и передачу информации серверу через шину данных Kafka.

NOTE

Предусмотрена возможность создания пользовательского агента, который будет собирать метрики с локального хоста, опрашивать сторонние сервисы и отправлять данные на сервер через REST API.

Web-клиент и **Мобильное приложение** – это клиенты, с помощью которых пользователи системы взаимодействуют с сервером через REST API. В роли клиентов могут выступать, например, web-приложения и клиенты для мобильных операционных систем Android и iOS.

NOTE

Для решения специфических задач можно создать индивидуального клиента, который будет работать с сервером через REST API.

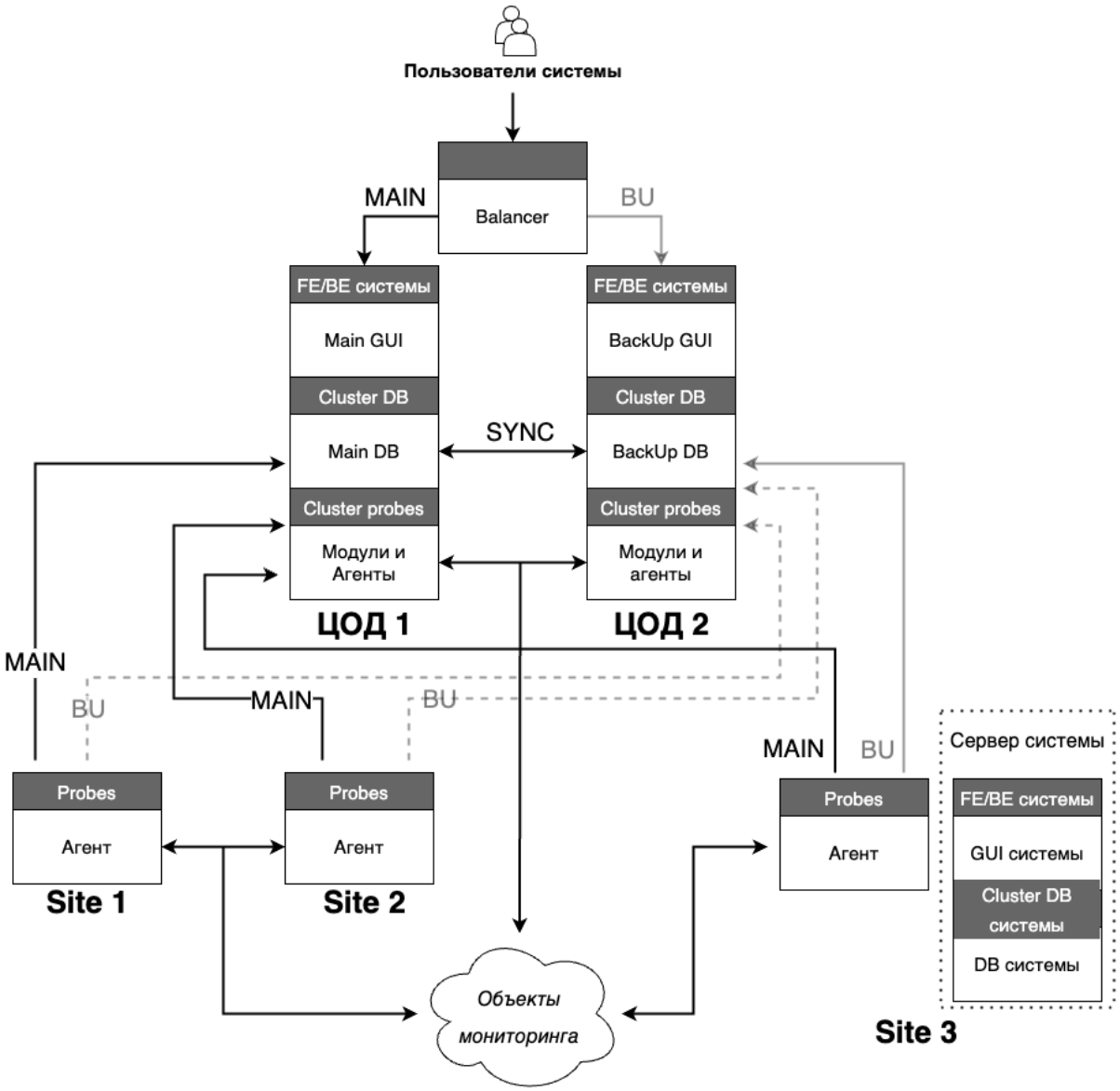
Внешние системы – это стороннее ПО, информационные системы или программные модули, которые могут взаимодействовать с сервером с помощью REST API.

Сервисы авторизации – наряду со встроенными механизмами, Центральный Пульт также поддерживает сторонние сервисы для авторизации пользователей в системе. Возможна интеграция с AD-сервером и Keycloak (см. LDAP и Интеграция с Keycloak).

Уведомления – сервер может уведомлять Пользователей об авариях и отправлять данные во Внешние Системы с использованием различных каналов, описанных в разделе Действия при смене состояний.

2.1.1.2. Отказоустойчивость и резервирование

Для обслуживания высокой вычислительной нагрузки система может быть реализована в отказоустойчивом исполнении с дублированием основных компонентов:



Производительность системы может быть увеличена следующими способами:

- Добавление ресурсов виртуальной инфраструктуры;
- Оптимизация параметров программного обеспечения.

2.1.1.3. Версии используемого и поддерживаемого ПО

- Keycloak 1.0 и выше;
- Драйвер MongoDB 6.16.0 (поддержка MongoDB от 4.0 до 8.0);
- NGINX 1.4 и выше;
- Node.js 20.11.x и выше;
- OpenTSDB 2.3 и выше;
- Redis 3.0 и выше;
- Socket.IO 4.0 и выше;
- TypeScript 4.7.4 и выше.
- KafkaJS 2.2.4

2.1.2. Сетевые доступы

В таблице ниже перечислены порты по умолчанию, необходимые для корректной работы системы Центральный Пульт:

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 системы Центральный Пульт	Серверы с сервисами системы Центральный Пульт	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 системы Центральный Пульт установлен агент, который получает SNMP Trap'ы.
443	TCP	HTTPS	Пользователи системы Центральный Пульт	Серверы с сервисами системы Центральный Пульт	Доступ к web-интерфейсу системы Центральный Пульт с шифрованием трафика.
3001	TCP	Wetty	Пользователи системы Центральный Пульт	Оборудование в инфраструктуре заказчика	Доступ к терминалам устройств через SSH и Telnet.
4242	TCP	OpenTSDB	Серверы с сервисами системы Центральный Пульт	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 системы Центральный Пульт	Серверы с Redis	Доступ к кэшу данных. Необходим при инсталляции Redis на отдельных серверах.

Порт	Протокол	Сервис	Инициатор соединения	Место назначения	Комментарий
8090	TCP	REST API	Пользователи, интеграционные модули и расширения	Серверы с сервисами системы Центральный Пульт	Взаимодействие с системой и данными через REST API.
8091	TCP	Notification Actor	Пользователи системы Центральный Пульт	Серверы с сервисами системы Центральный Пульт	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 системы Центральный Пульт	Отображение в web-интерфейсе приложенных к объектам документов и фоновых изображений.
8428	TCP	VictoriaMetrics	Серверы с сервисами системы Центральный Пульт	Серверы с VictoriaMetrics	Доступ к базе данных хранения временных рядов для построения графиков. Необходим при инсталляции VictoriaMetrics на отдельных серверах.
9092	TCP	Kafka	Серверы с сервисами системы Центральный Пульт 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 системы Центральный Пульт	Серверы с MongoDB	Доступ к базе данных основной информации об объектах и пользователях. Необходим при инсталляции MongoDB на отдельных серверах.

2.1.3. Файлы конфигурации

Функционал системы Центральный Пульт можно настроить в следующих конфигурационных файлах, которые находятся на сервере:

- config/server-config.html
- config/client-config.html
- config/log-config.html

NOTE

Некоторые параметры конфигурации можно изменить через REST API, с помощью соответствующих запросов для конфигурации сервера и Web-приложения.

Также, некоторые системные настройки находятся в MongoDB. Изменить их можно в окне конфигурации системы.

2.1.3.1. 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.**password**
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 системы Центральный Пульт 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 системы Центральный Пульт.

```
"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 журналирования 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 системе Центральный Пульт;

-

false – разрешена авторизация всех новых LDAP-пользователей,

- true – разрешена авторизация только тех новых LDAP-пользователей, для которых заготовлены группы в системе Центральный Пульт.

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 системе Центральный Пульт;

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

По умолчанию – `false` – при авторизации LDAP-пользователей в системе Центральный Пульт не создаются новые группы.

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

logdb

В этом разделе задаются настройки базы данных для записей журнала безопасности.

WARNING

Этот раздел требуется для включения интерфейса журнала безопасности. Этот раздел не задан по умолчанию.

```
"logdb": {
  "url": "mongodb://localhost:27017/logs",
  "collection": "securityLogs"
}
```

logdb.**url**
Путь до базы данных MongoDB, в которой должны храниться записи журнала безопасности.

logdb.**collection**
Название коллекции записей журнала безопасности.

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.

tsdb_extensions

В этом разделе задаются параметры конфигураций дополнительных баз временных рядов.

```
"tsdb_extensions": [
    {
       "name": "influx",
       "enabled": true,
       "config": {}
       "path": <path to DB client file>
    }
],
```

tsdb_extensions.**name**
Имя дополнительной базы временных рядов;

Поддерживаемые значения:

- `influx` – InfluxDB
- `victoria` – VictoriaMetrics

tsdb_extensions.**enabled**
Запись исторических данных в дополнительную базу временных рядов;

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

По умолчанию – `true` – запись включена.

openTsdb.**config**
Задает конфигурацию для подключения к БД. Настройки конфигурации для каждой поддерживаемой БД описаны в соответствующих описаниях интеграции:

- InfluxDB
- VictoriaMetrics

openTsdb.**path**
Задает абсолютный путь до файла клиента БД. Используется в том случае, если требуется подключить ещё какой-либо сторонний модуль для работы с временными рядами.

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 Центральный Пульт 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 журналирования 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",
    "body_limit": "100kb",
    "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.**body_limit**
Ограничение на размер тела запроса к REST серверу.

Этот параметр можно задавать количеством байт, или строкой формата `<размер><единица измерения>` (например "10kb" или "5mb").

При превышении этого ограничения при запросе через Web-интерфейс, пользователю будет показано уведомление "Объект запроса слишком велик".

rest_server.**colorize_log**
Включает цветную раскраску журнала;

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

По умолчанию – `false` – раскраска выключена.

rest_server.**debug**
Включает debug-режим для журналирования 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.**active_keys_expire_check_period**
Период проверки устаревания данных от агента.

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.**block_by_ip**
Заблокировать пользователя по IP при определённом количестве неудачных попыток входа.

Если пользователь заблокирован, то при входе ему будет показана следующая ошибка:

```
Ошибка! (38) Слишком много неудачных попыток входа в систему, повторите попытку через
несколько секунд
```

По умолчанию этот раздел не определён, функция отключена.

```
"block_by_ip": {
  "enabled": true,
  "attempts": 3,
  "block_period": 30000
}
```

server.block_by_ip.**enabled**

Включён ли функционал блокировки пользователя по IP.

server.block_by_ip.**attempts**
Количество попыток, которые приводят к блокировке IP адреса.

server.block_by_ip.**block_period**
На какое время заблокировать IP пользователя (в миллисекундах).

server.**colorize_log**
Включает цветную раскраску журнала;

- 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 журналирования 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).

WARNING

Если установить слишком большое значение этой переменной, то пользователь может получить ошибку `504 Timeout`, даже если операция заняла меньше времени, чем указано в этой переменной. Это происходит за-за конфликта с таймаутами, указанными в nginx. В этом случае нужно изменить настройки nginx следующим образом:

```
server {
    # Настройки таймаутов для проксирования(для длительных операций)(180s=3m)
        proxy_read_timeout 180s;
        proxy_connect_timeout 180s;
        proxy_send_timeout 180s;
}
```

Эту конфигурацию нужно изменять в файле `/etc/nginx/sites-available/saymon` для стандартной установки и в `/data/nginx/conf.d/saymon.conf`, если сервер системы Центральный Пульт установлен в Docker.

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.**state_map_update_period**
Период обновления таблицы состояний.

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.

TIP

Система Центральный Пульт использует Nodemailer для отправки писем. Более подробную информацию о конфигурации почтового сервера можно найти в документации Nodemailer

```
"email": {
  "disabled": false,
  "transport": {
    "auth": {
      "user": "user@example.com",
      "pass": "password"
    },
    "host": "example.com",
    "port": "587",
    "requireTLS": true,
    "secure": false,
    "tls": {
      "minVersion": "TLSv1.2",
      "ciphers": "ECDHE-RSA-AES256-GCM-SHA384",
      "rejectUnauthorized": true
    }
  },
  "fields": {
    "from": "user@example.com"
  },
  "max_json_length": 1000
}
```

server.email.**disabled**
Выключает отправку почтовых уведомлений. Этот параметр можно использовать для тестирования конфигурации или временного отключения email-уведомлений.

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

По умолчанию – `true` – отправка выключена.

server.email.**max_json_length**
Максимальная длина JSON-объекта в символах, который может быть отправлен SMTP-сервером. Этот параметр может использоваться для ограничения размера данных, передаваемых в письме.

По умолчанию – `1000` – 1000 символов.

server.email.**fields**
Дополнительные параметры отправки писем.

server.email.fields.**from**
Адрес отправителя по умолчанию, который будет использоваться, если не указан другой.

server.email.fields.**to**
Адрес получателя по умолчанию.

server.email.**transport**
Параметры подключения к SMTP-серверу. Эти параметры передаются Nodemailer для установки соединения с сервером.

server.email.transport.**auth**
Данные для аутентификации пользователя.

server.email.transport.auth.**user**
Логин пользователя (email-адрес) для аутентификации на SMTP-сервере.

server.email.transport.auth.**pass**
Пароль пользователя для аутентификации на SMTP-сервере.

server.email.transport.**host**
Адрес SMTP-сервера.

server.email.transport.**port**
Порт для подключения к SMTP-серверу.

TIP

Значение `587` обычно используется для STARTTLS-соединений.

server.email.transport.**requireTLS**
Требовать использование TLS при установке соединения:

- `true` – Nodemailer будет требовать использование TLS,
- `false` – Nodemailer не будет требовать использование TLS.

server.email.transport.**secure**
Должно ли соединение сразу устанавливаться в защищённом режиме.

- `false` – соединение будет установлено в незащищённом режиме, а затем будет переключено на TLS (STARTTLS),
- `true` – соединение сразу использует TLS.

Значение по умолчанию зависит от используемого порта.

server.email.transport.**service**
Встроенный в коннектор набор служб.

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

По умолчанию – **"Gmail"**.

server.email.**tls**
Дополнительные параметры для настройки TLS-соединения.

server.email.tls.**minVersion**
Минимально допустимая версия TLS (например, `TLSv1.2`).

server.email.tls.**ciphers**
Перечисление шифров (например, `"ECDHE-RSA-AES256-GCM-SHA384"`) для установления защищённого соединения.

server.email.tls.**rejectUnauthorized**
Отклонять соединения с недостоверными сертификатами.

- `false` – SMTP-сервер отклоняет незащищённые соединения,
- `true` – SMTP-сервер принимает незащищённые соединения.

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.**csrf_token_enabled**
Включает защиту от CSRF/XSRF.

- false – сервер не использует защиту от CSRF атак,
- true – сервер использует защиту от CSRF .

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 журналирования 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 системы Центральный Пульт ниже 3.12.86, при переходе на новую версию необходимо выполнить следующее:

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 журнала с управляющими ANSI-символами;

- true – журнал записывается с управляющими ANSI-символами;
- false – журнал записывается без управляющих ANSI-символов.

По умолчанию – `true` – журнал записывается 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 журнал;

- 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 системы Центральный Пульт 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 системы Центральный Пульт 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 Zabbix API (англ.).

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 Zabbix API (англ.).

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 Zabbix API (англ.).

2.1.3.2. Конфигурация Web-приложения

Параметры конфигурации web-приложения системы Центральный Пульт 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);
- `./backend/server/saymon-client-default.yaml` – параметры конфигурации web-приложения по умолчанию.

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

Также, параметры конфигурации могут храниться в файле `/etc/saymon/client-config.js`. Этот способ конфигурации устарел и не должен быть использован с новыми версиями web-приложения.

agentInstallerUrl

Ссылка для скачивания установочного скрипта для агента.

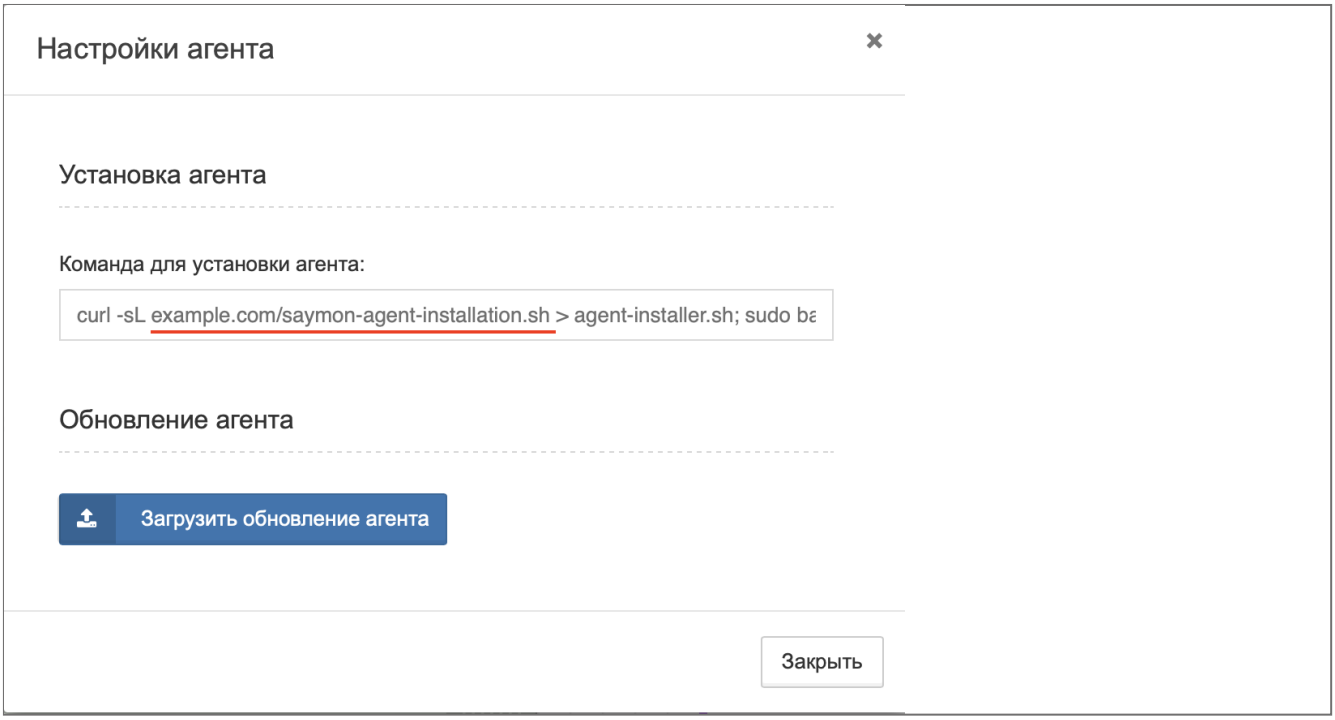
YAML

```
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

```
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

```
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

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

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 системы Центральный Пульт 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

```
canEditObjectStyle: true
```

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 системы Центральный Пульт через Comet-сервер.

YAML

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

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

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

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

```
collapseSections: true
```

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

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

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.

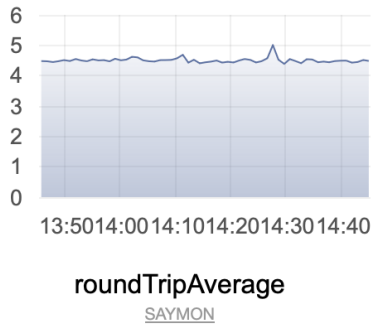
defaultClassImage.<number>.**dim**
Размер изображения (в пикселях, [ширина, высота]).

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

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

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

```
{
  "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 системы Центральный Пульт.

YAML

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

JSON

```
"documentation": {
  "main": {
      "root": "https://docs.saymon.tech",
      "ru": "https://docs.cpult.ru"
  },
  "notificationTemplates": {
      "root": "xref:web-interface:system-configuration-window/notification-
templates.html",
      "ru": "xref: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 системы Центральный Пульт.

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

```
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.

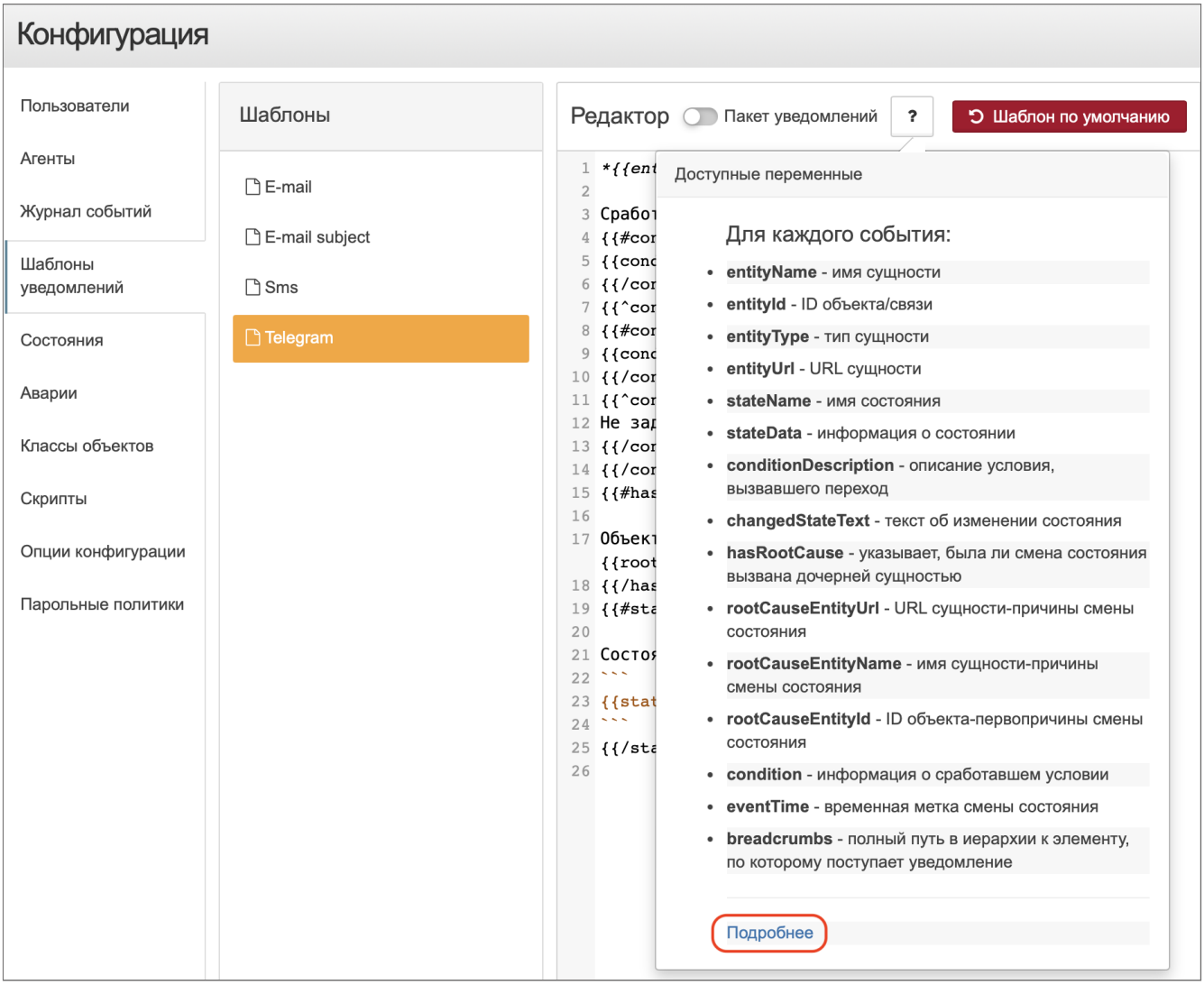
По умолчанию:

- 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 web-interface:configuration/notification-templates.html.

enableAnalytics

Включает функционал модуля аналитики;

YAML

```
enableAnalytics: false
```

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

```
enableJiraTrigger: false
```

JSON

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

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

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

enableConditionalIncidents

Включает функционал генерации аварий;

YAML

```
enableConditionalIncidents: false
```

JSON

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

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

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

enableSmsTrigger

Включает использование SMS-уведомлений;

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

```
enableSnmpTranslate: false
```

JSON

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

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

По умолчанию – **false** – трансляция выключена.

enableVoiceCallTrigger

Включает использование голосовых вызовов;

YAML

```
enableVoiceCallTrigger: false
```

JSON

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

- false – голосовые вызовы выключены,
- true – голосовые вызовы включены.

По умолчанию – **false** – голосовые вызовы выключены.

experimental

Включает экспериментальный функционал в различных частях системы.

YAML

```
experimental: false
```

JSON

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

forceEmptyStandardView

Список классов для которых пользовательский интерфейс будет по умолчанию открывать стандартный вид, даже при отсутствии дочерних объектов.

YAML

```
forceEmptyStandardView:
  - 1
  - 3
  - 13
  - 43
```

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

```
geoMap:
  serverName: openstreetmap.org
  initialPosition:
    - 30.341306
    - 59.930089
  initialZoom: 8
```

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

```
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`.

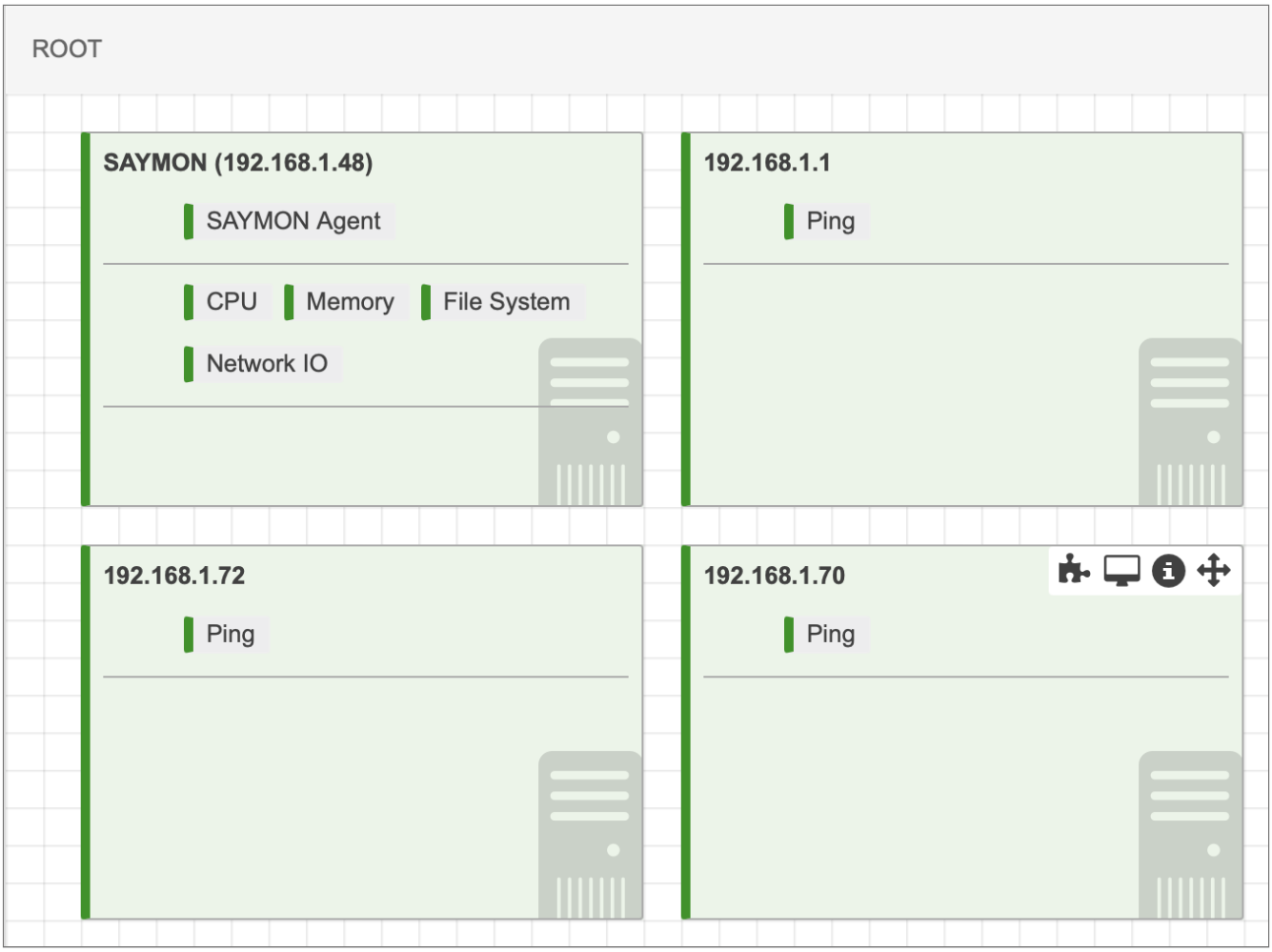
По умолчанию – openstreetmap.org

Также можно использовать локальный OSM-сервер.

geoMap.**skipModalView**
При нажатии на объект в режиме Геокарты пропустить модальное окно и сразу открывать информацию в виде по умолчанию для этого объекта.

grid

В разделе задаются настройки сетки для выравнивания объектов в стандартном виде.



YAML

```
grid:
  dim: 20
  color: rgba(128, 128, 128, 0.3)
  border: 4
```

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).

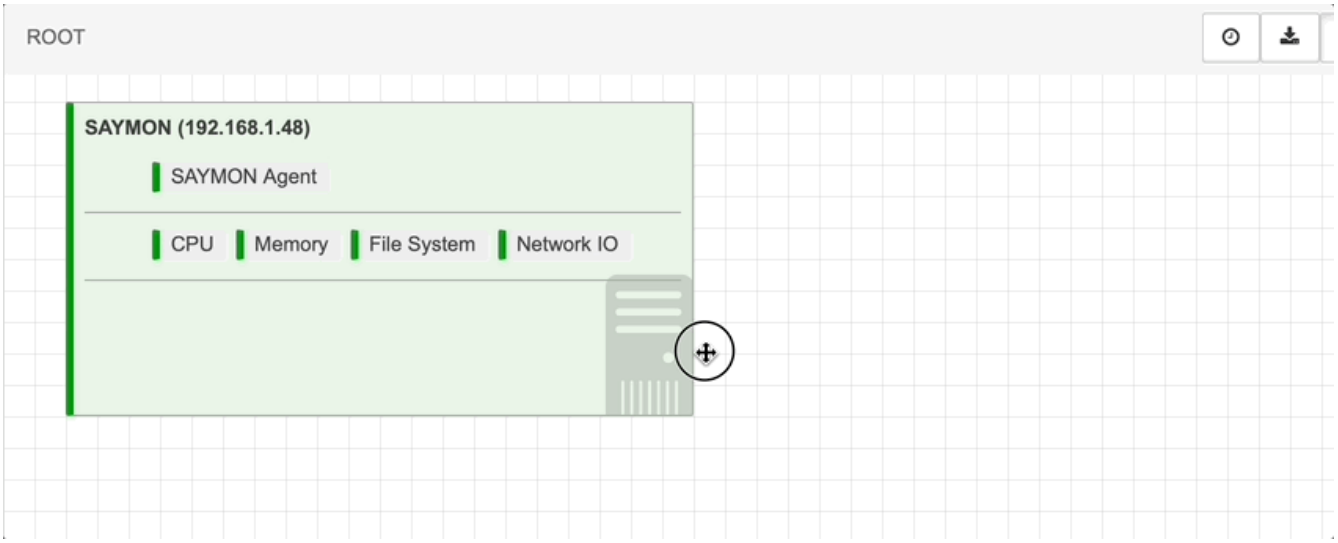
По умолчанию – **20**.

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`):



По умолчанию – **4**.

gridView

В разделе задаются параметры отображения объектов в виде сетки.

YAML

```
gridView:
  cols: 6
  colWidth: 240
  rowHeight: 90
  gutterSize: 5
```

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

```
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
  pagination: false
  paginationLimit: 10
```

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,
    "pagination": false,
    "paginationLimit": 10
  }
}
```

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;

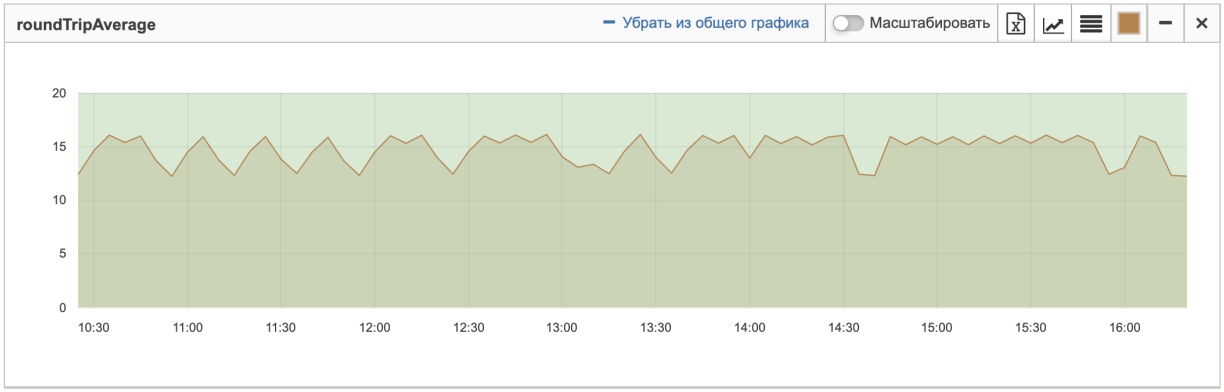
- `m` – минуты;
- `h` – часы;
- `d` – дни.

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.

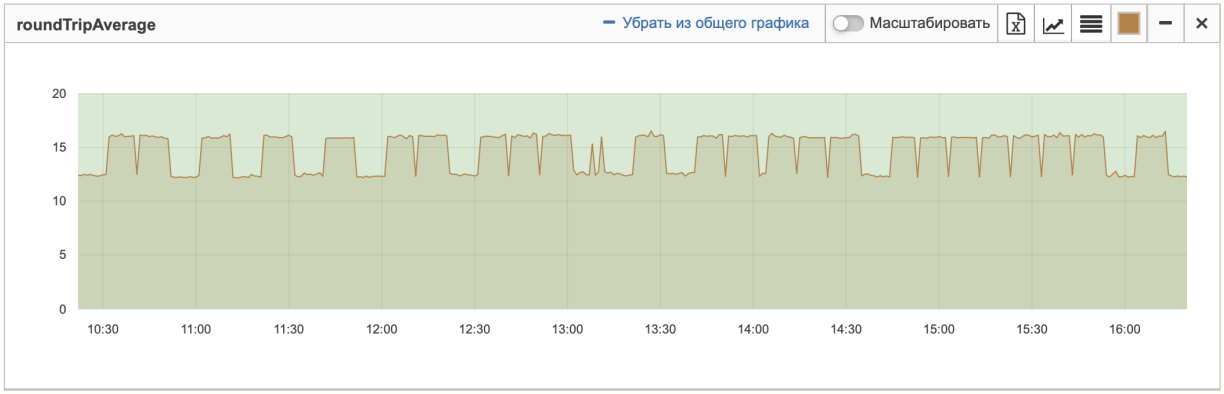
```
history:
  downsampling:
    - period: 6-hours
      dsp: 5m-avg
```



"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.

history.pagination

Включение постраничного вывода на графиках.

- false – постраничный вывод на графиках выключен,
- true – постраничный вывод на графиках включена.

По умолчанию – **false** – постраничный вывод на графиках выключен.

history.paginationLimit

Количество графиков на одной странице.

По умолчанию – **10**.

hideConnectionStatusNotifications

Скрывает оповещения о статусе соединения с Comet-сервером;

YAML

```
hideConnectionStatusNotifications: false
```

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

```
hideResponseErrorNotifications: false
```

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.

hiddenPropertyTypes

Список типов свойств, скрываемых из интерфейса.

TIP

Свойства скрываются только из web-интерфейса системы. Скрытые свойства можно возвращать и обновлять соответствующими запросами REST API.

YAML

```
hiddenPropertyTypes:
  - 101
  - 126
  - 310
  - 514
```

JSON

```
{
  "hiddenPropertyTypes": [
    101,
    126,
    310,
    514
  ]
}
```

incidents

В этом разделе настраиваются дополнительные поля для списков активных аварий и исторических аварий.

YAML

```
incidents:
  additionalFields:
    - owner.properties.address
    - owner.class_id
    - updated
  additionalFieldNames:
    - IP-address
    - Object Class ID
```

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.

По умолчанию, в системе доступен только русский язык. Для включения других языков, их все нужно в явном виде указать в файле конфигурации.

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 интерфейса системы Центральный Пульт при выборе языка интерфейса.

languages.<id>.**short**
Идентификатор иконки флага.

Иконка отображается в меню текущего пользователя интерфейса системы Центральный Пульт при выборе языка интерфейса.

 – es,

 – gb,

 – it,

 – ru,

 – us.

NOTE

Если указать несуществующий или некорректный идентификатор, то будет использован флаг .

monitoring

В этом разделе задаются настройки сенсоров мониторинга, такие как скрытие стандартных мониторинговых сенсоров из интерфейса и привязка словарей для мониторинга к сенсорам.

YAML

```yaml
monitoring:
  standardTasks:
    - ping
    - snmpGet
  taskDictionaries:
    ftp: # Стандартная задача для мониторинга FTP соединения
      - dictionary: connection-credentials # Словарь учётных данных подключений
        mapping:
          user: FtpLogin # Связь поля user из словаря с параметром FtpLogin задачи
          password: FtpPassword
    jmx:
      - dictionary: connection-credentials
        mapping:
          user: JmxLogin
          password: JmxPassword
    snmpGet:
      - dictionary: snmp-credentials
        mapping:
          communityRead: SnmpCommunity
          user: SnmpUser
          authType: SnmpAuth
          authPassword: SnmpAuthPass
          privType: SnmpPriv
          privPassword: SnmpPrivPass
```

JSON

```
{
    "monitoring": {
        "standardTasks": [
            "ping",
            "snmpGet"
        ],
        "taskDictionaries": {
            "ftp": [
                {
                    "dictionary": "connection-credentials",
                    "mapping": {
                        "user": "FtpLogin",
                        "password": "FtpPassword"
                    }
                }
            ],
            "jmx": [
                {
                    "dictionary": "connection-credentials",
                    "mapping": {
                        "user": "JmxLogin",
                        "password": "JmxPassword"
                    }
                }
            ],
            "snmpGet": [
                {
                    "dictionary": "snmp-credentials",
                    "mapping": {
                        "communityRead": "SnmpCommunity",
                        "user": "SnmpUser",
                        "authType": "SnmpAuth",
                        "authPassword": "SnmpAuthPass",
                        "privType": "SnmpPriv",
                        "privPassword": "SnmpPrivPass"
                    }
                }
            ]
        }
    }
}
```

monitoring.**standardTasks**
Список идентификаторов отображаемых стандартных сенсоров. Список идентификаторов можно посмотреть в разделе Список идентификаторов стандартных сенсоров.

Если этот раздел не задан в файле конфигурации, то в интерфейсе отображаются все стандартные сенсоры.

monitoring.**taskDictionaries**
Привязка словарей для мониторинга к сенсорам мониторинга (в том числе и пользовательских).

Сенсоры задаются следующим образом:

YAML

```
monitoring:
  taskDictionaries:
    ftp:
      - dictionary: connection-credentials
        ...
    jmx:
      - dictionary: connection-credentials
        ...
    snmpGet:
      - dictionary: snmp-credentials
        ...
    custom/probe.sh:
      - dictionary: custom-probe-dictionary
        ...
      - dictionary: 672c7063066d7b194b37563c
        ...
```

JSON

```
{
    "monitoring": {
        "taskDictionaries": {
            "ftp": [
                {
                  "dictionary": "connection-credentials",
                  ...
                }
            ],
            "jmx": [
                {
                  "dictionary": "connection-credentials",
                    ...
                }
            ],
            "snmpGet": [
                {
                  "dictionary": "snmp-credentials",
                    ...
                }
            ],
            "custom/probe.sh": [
                {
                  "dictionary": "custom-probe-dictionary",
                    ...
                  "dictionary": "672c7063066d7b194b37563c",
                    ...
                }
            ]
        }
    }
}
```

monitoring.taskDictionaries.**dictionary**
Идентификатор связываемого с сенсором словаря. Это единственный обязательный параметр. Если этот параметр не указан, система проигнорирует эту запись.

К каждому сенсору может быть привязано больше одного словаря.

monitoring.taskDictionaries.**name**
Альтернативное наименование словаря, которое будет отображено для него в пользовательском интерфейсе выбора словаря для сенсора. Может быть использовано, если собственное наименование словаря не подходит, а также для локализации на поддерживаемые системой языки.

YAML

```
name:
  - en: My dictionary
  - ru: Мой словарь
  - ..: ...
```

JSON

```
{
    "name": [
        {
            "en": "My dictionary"
        },
        {
            "ru": "Мой словарь"
        },
        {
            "..": "..."
        }
    ]
}
```

TIP

При отсутствии необходимого значения для текущего выбранного языка, будет использовано первое значение найденное в этом списке.

monitoring.taskDictionaries.**map_all**
Автоматическое назначение всех полей словаря в соответствующие (по наименованию) свойства сенсора.

- `true` – система автоматически назначает соответствующие поля словаря,
- `false` – система использует поля, определённые в поле `monitoring.taskDictionaries.mapping`.

По умолчанию – `false` – система использует поля, определённые в поле `monitoring.taskDictionaries.mapping`.

monitoring.taskDictionaries.**mapping**
Определяет карту связи полей из словаря с параметрами сенсора. Если данный параметр не указан, то производится связь всех именованных полей словаря с идентичными по названию параметрами сенсора.

Используется, если имена параметров сенсора отличаются от наименований полей в словаре.

monitoring.taskDictionaries.**exclude**
Список полей словаря, которые не будут связаны с параметрами сенсора.

Данный параметр необходим только в тех случаях, когда надо исключить пересечение наименований при использовании нескольких словарей в одном сенсоре и одновременно включённом параметре `map_all`.

monitoring.taskDictionaries.**template_for**
Имя свойства сенсора, формирование которого будет определять текстовый шаблон выбранный из данного словаря.

millisecondMonitoring

Включает обновление данных каждую миллисекунду.

YAML

```
millisecondMonitoring: false
```

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

```
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

```
title: "SAYMON"
```

JSON

```
{
  "title": "SAYMON",
  ...
}
```

По умолчанию – **SAYMON**.

themes

Этот параметр отвечает за настройку тем.

YAML

```
themes:
  default:
    name: Light
  dark:
    name: Dark
```

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

```
themes:
  my-custom-theme:
    name: Custom Theme
```

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

```
tree:
  filterSubmit: true
```

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

```
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).

2.1.3.3. Конфигурация журналирования сервера

Параметры журналирования сервера системы Центральный Пульт задаются в файле `/etc/saymon/logger.json`.

Параметры журналирования

Ниже приведён пример настроек журналирования для сервера системы Центральный Пульт.

```
{
  // Категории журнала
  "categories": {
    "Default": "Info"
  },
  // Каналы журналирования
  "channels": {
    // Канал 'default' предназначен для сообщений, которые не подходят к другим каналам (Общий
канал)
    "default": [
      {
        "type": "console",
        "options": {
          "level": "debug",
          "stderrLevels": [
            "error"
          ],
          "colorize": false,
          "prettyPrint": true
        }
      }
    ],
    // Канал 'session' предназначен для сообщений о сессии пользователей (Частный канал)
    "session": [
      // Пример 1: отправка записей журнала на сервер по протоколу UDP
      // {
      //   type: 'udp',
      //   options: {
      //     server: 'localhost:514'
      //   }
      // },
      //
      // Пример 2: отправка записей журнала на сервер Kafka
      // {
      //   type: 'kafka',
      //   options: {
      //     server: 'localhost:9092',
      //     topic: 'LOG_SESSION'
      //   }
      // }
    ],
    // Канал 'audit' предназначен для сообщений об изменении параметров системы (Частный
канал)
    "audit": [],
    // Канал 'faults' предназначен для сообщений об ошибках при доступе к системе (Частный
канал)
    "faults": []
  }
}
```

categories

```
{
  "categories": {
    "Default": "Info",
    "MyActor": "Error",
    "MyOtherActor": "Debug"
  }
  ...
}
```

Раздел/Параметр	Описание
categories.**<название-категории>**	Указание минимального уровня журналирования для указанной категории. **NOTE** Существует специальная категория, `"Default"`, которая настраивает уровень журналирования по умолчанию. Возможные уровни журналирования: • "Silent", • "Info", • "Warn", • "Error", • "Debug"

channels
В этом разделе можно определить настройки каналов журналирования.

Журналирование в системе Центральный Пульт разделяет все записи журналов на каналы. Каждый канал представляет собой определённый способ обработки записей журналов.

Сообщения системы безопасности можно отправлять в разные каналы. Каждый канал может содержать несколько транспортных модулей (механизмов доставки в соответствующую систему обработки, например `udp` или `kafka`) для доставки сообщения одновременно в несколько систем.

Например, сообщения об ошибках можно отправлять в канал "faults", который будет их перенаправлять одновременно в консоль и во внешнюю систему (например в syslog сервер).

```
{
  "channels": {
    "faults": [
      {
        "type": "console",
        "options": {
          "level": "debug"
        }
      },
      {
        "type": "udp",
        "options": {
          "server": "localhost:514"
        }
      }
    ]
  }
}
```

В системе определены 4 тематических канала:

- `session` - информация об открытии/закрытии пользовательской сессии,
- `audit` - информация об изменениях в системе,
- `faults` - информация об ошибках при доступе к системе.
- `default` – канал для сообщений, которые не подходят в другие каналы,

По умолчанию, в системе определён только канал `default`. Этот канал отправляет сообщения в консоль на сервере.

Все сообщения, отправленные в канал, который не был определён, игнорируются.

Для каждого канала можно определить несколько транспортных модулей для одновременной доставки сообщений в несколько систем хранения и обработки.

```
channels {
    ...
    "session": [
      // Отправка записей журнала на сервер через UDP
      {
        "type": "udp",
        "options": {
          "server": "localhost:514"
        }
      },

      // Отправка записей журнала в Kafka
      {
        "type": "kafka",
        "options": {
          "server": "localhost:9092",
          "topic": "LOG_SESSION"
        }
      }
    ],
    ...
}
```

NOTE

По умолчанию, в системе определён канал `default`. Указывать его настройки в файле конфигурации нужно только если требуется изменить его стандартное поведение (отправка сообщений в консоль на сервере).

Раздел/ Параметр	**Описание**
channels.<назвзание-канала>.**type**	Тип транспортного модуля. Возможные типы модулей: • null (отключает журналирование), • udp, • kafka, • console.
channels.<назвзание-канала>.**options**	Настройки транспортного модуля. Зависят от его типа.

common

```
"common": {
  "resolveRemoteIp": true
},
```

Раздел/Параметр	Описание
common.**resolveRemoteIp**	Если выставить этот параметр в значение `true`, то система будет автоматически определять имя клиента через DNS и включать его к сообщениям Журнала безопасности. **WARNING** Эта настройка может повлиять на производительность системы журналирования и должна включаться только при особой необходимости. По умолчанию этот функционал отключён.

udp

Раздел/ Параметр	Описание
options.**server**	Адрес UDP сервера. Система принимает следующие форматы адреса: • <ip-address|hostname>:<port> • <ip-address|hostname> • <port>

kafka

Раздел/ Параметр	Описание
options.**server**	Адрес сервера Kafka. Система принимает следующие форматы адреса: • <ip-address|hostname>:<port> • <ip-address|hostname> • <port>
options.**topic**	Топик Kafka, в который отправляется событие.

console

Раздел/Параметр	Описание
options.**level**	Минимальный уровень сообщений, которые выводятся на консоль. По умолчанию `'debug'`
options.**stderrLevels**	Список уровней журналирования, сообщения на которых нужно выводить в stderr, а не в stdout. Например `['error', 'debug', 'info']`. По умолчанию, список пустой.
options.**colorize**	Выделять уровни записей журнала цветом. По умолчанию `false`.
options.**prettyPrint**	Форматировать сообщения в консоли. По умолчанию `true`.

Параметры ротации логов

Сервер системы Центральный Пульт производит ротацию логов с помощью `logrotate`.

Конфигурация ротации журналов сервера системы Центральный Пульт выполняется в файле `/etc/logrotate.d/saymon`.

Конфигурация `logrotate` имеет следующую структуру:

```
<путь до файлов журнала> {
    параметры конфигурации
}
```

Ниже приведена стандартная конфигурация `logrotate` на сервере для системы Центральный Пульт и MongoDB.

Журнал системы Центральный Пульт

```
/var/log/saymon/*.log {
    daily
    missingok
    rotate 14
    compress
    delaycompress
    notifempty
    copytruncate
}
```

Журнал MongoDB

```
/var/log/mongodb/*.log {
    daily
    rotate 30
    compress
    dateext
    missingok
    notifempty
    sharedscripts
    copytruncate
}
```

В таблице указаны используемые параметры `logrotate`:

Параметр	Описание
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`.
delaycompress	Не архивировать последний журнал до следующего цикла ротации.
notifempty	Не производить ротацию журнала, если он пуст.
copytruncate	Писать журнал в один файл, урезая его после каждого шага ротации.
sharedscripts	Выполнять скрипты `prescript` и `postscript` только один раз для всего процесса ротации, а не для каждого журнала.

2.2. Инструкции

2.2.1. 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`:

   YAML

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

   JSON

```
{
  "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
```

2.2.2. Настройка пользовательских сенсоров мониторинга

В качестве сенсоров мониторинга можно использовать собственные скрипты и передавать им аргументы, настраиваемые через web-интерфейс.

Каждая проверка состоит из двух файлов:

- исполняемый файл проверки (скрипт или бинарный файл);
- файл конфигурации, описывающий web-интерфейс секции Мониторинг 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-agent/custom_tasks/mytask.sh
/opt/saymon-agent/custom_tasks/mytask.sh.conf
```

TIP

Файл пользовательской проверки должен быть исполняемым и быть доступен пользователю `saymon`. В противном случае, в таблице Данные будет отображаться ошибка `error=13, Permission denied`.

Исполняемые файлы пользовательских сенсоров можно группировать в подкаталогах с любым уровнем вложенности:

```
/opt/saymon-agent/custom_tasks/cisco/task_1.sh
/opt/saymon-agent/custom_tasks/cisco/task_1.sh.conf
/opt/saymon-agent/custom_tasks/cisco/task_2.sh
/opt/saymon-agent/custom_tasks/cisco/task_2.sh.conf
/opt/saymon-agent/custom_tasks/hp/task_3.sh
/opt/saymon-agent/custom_tasks/hp/task_3.sh.conf
/opt/saymon-agent/custom_tasks/hp/task_4.sh
/opt/saymon-agent/custom_tasks/hp/task_4.sh.conf
```

В web-интерфейсе данные проверки будут группироваться и отображаться с учётом подпапок:

```
cisco - task_1
cisco - task_2
hp - task_3
hp - task_4
```

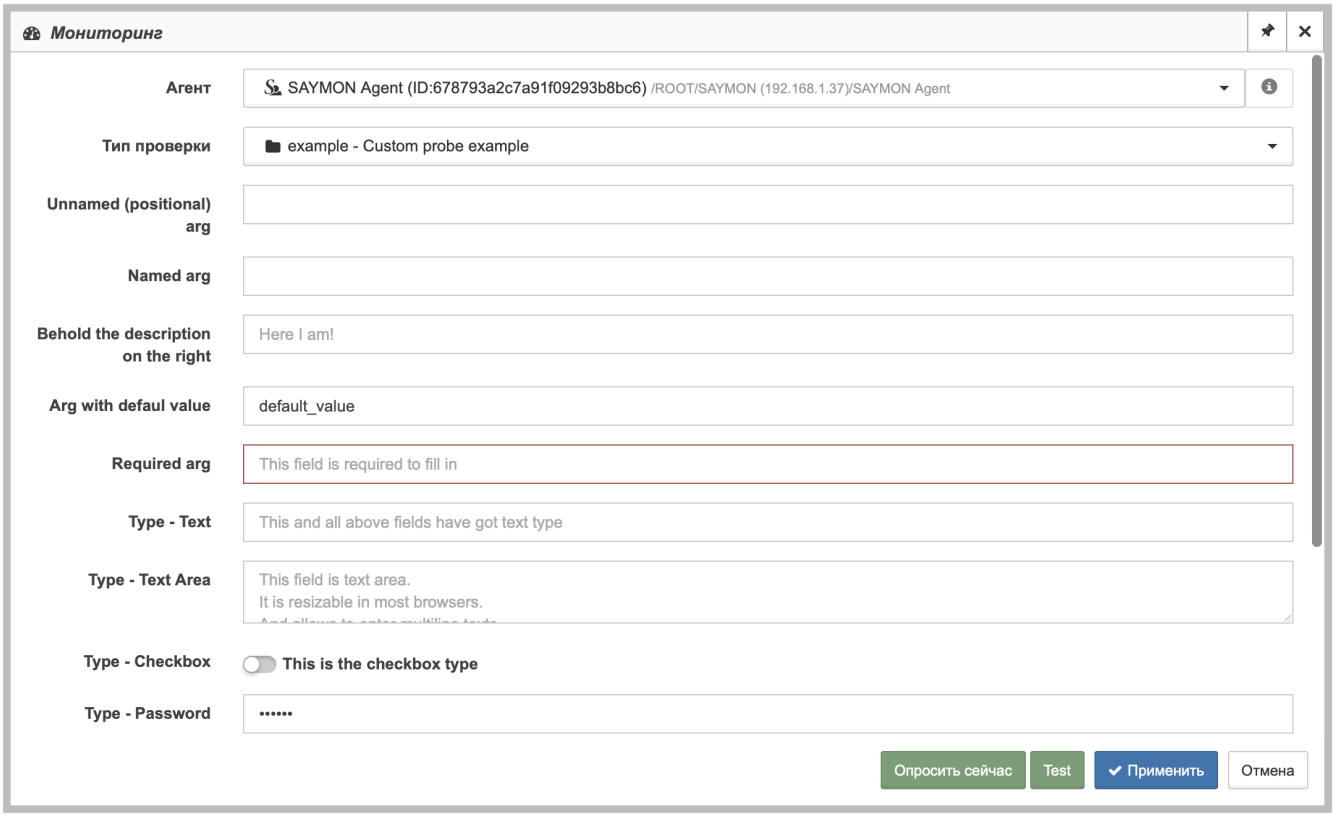


Рис. 1. Отображение примера пользовательского сенсора

При добавлении или модификации проверок все изменения применяются автоматически – достаточно обновить страницу в браузере.

CAUTION

Идентификаторы сенсоров, которыми оперируют сервер и агенты, создаются автоматически и эквивалентны именам исполняемых файлов с частью пути от сконфигурированной папки. Например: **cisco/task_1**.

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

С перечнем идентификаторов стандартных сенсоров можно ознакомиться в статье Скрытие неиспользуемых сенсоров мониторинга.

Идентификаторы сенсоров чувствительны к регистру символов.

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.

2.2.2.1. Каталог для пользовательских сенсоров

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

Необходимо:

Добавить в файл секцию **monitoring.**

2. В параметре **custom_tasks_path** задать полный путь до папки с проверками.

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

3. Перезапустить сервер:

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

2.2.2.2. Параметры пользовательского сенсора

Параметры сенсора задаются в JSON-файле, именуемом как **имя_исполняемого_файла.conf**. Например, для сенсора **mySensor.sh** файл настроек параметров должен называться **mySensor.sh.conf**.

Конфигурационный файл должен располагаться в одном каталоге с исполняемым файлом.

CAUTION

Сенсоры без конфигурационного файла игнорируются.

Файл настроек параметров может содержать следующий набор полей:

Поле	Описание
title	Имя сенсора, отображаемое в web-интерфейсе в списке проверок. Опциональное поле. По умолчанию в качестве имени проверки отображается идентификатор сенсора, генерируемый сервером из пути и имени скрипта: • путь к файлу: *<путь до папки с проверками>***/cisco/task_1.sh** • отображаемое имя: **cisco - task_1**
icon	Иконка из набора Font Awesome, отображаемая рядом с именем проверки. Опциональное поле. По умолчанию отображается – 'fa fa-cubes fa-fw'
args	Настройки аргументов, которые задаются пользователем через web-интерфейс и передаются исполняемому файлу.

Поле	Описание
args.**id**	Имя аргумента, передаваемое исполняемому файлу. Опциональное поле.
args.**default**	Значение аргумента по умолчанию, подставляемое в соответствующее поле при создании проверки. Опциональное поле.
args.**description**	Описание аргумента, отображаемое в web-интерфейсе. Опциональное поле.
args.**name**	Название аргумента, отображаемое в web-интерфейсе.
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). **TIP** 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).

Поле	Описание
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.**required**	Признак обязательности заполнения значения аргумента. • **true** – значение обязательно, • **false** – значение опционально. Опциональное поле. По умолчанию – **false**.
args.**type**	Тип значения аргумента. Опциональное поле. По умолчанию – **text**.
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).

2.2.2.3. Пример пользовательского сенсора

Для демонстрации работы пользовательского сенсора и различных типов аргументов мы подготовили два файла.

Разместите их в каталоге пользовательских проверок, создайте объект с проверкой в web-интерфейсе и попробуйте поменять значения аргументов.

Исполняемый файл проверки выводит все переданные в него аргументы:

hello_world.sh

```
#!/bin/bash
#
# Use this script as the custom monitoring sensor
# to check or test arguments configuration file.

i=0
echo "Custom probe got the following args:"
for arg in $*
do
  i=$((i + 1))
  echo -ne $arg; echo -ne " "
done

if [ $i = 0 ]
then
  echo "(no args)"
  exit 2
fi
```

В этом конфигурационном файле приведены примеры для всех типов аргументов:

hello_world.sh.conf

```
{
    "title": "Custom probe example",
    "icon": "fa fa-folder fa-fw",
    "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 defaul 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",
```

Загрузка файла

Аргумент типа `file` позволяет загружать небольшой (до 16Мбайт) произвольный файл, привязав его к параметру свойства пользовательской проверки.

```
{
    ...
    "args": [
        {
            "name": "Type - File",
            "description": "This argument allows you to upload a file up to 16 MBs",
            "id": "--file",
            "type": "file"
        },
    ]
    ...
}
```

Загруженный файл будет храниться в отдельной коллекции в MongoDB (`fileStorage`). Доступ к этому файлу можно будет получить по следующей ссылке:

```
http://<server-hostname>/node/api/files/<file-id>
```

WARNING

Для доступа к файлу по ссылке требуется авторизация. При этом, для доступа к файлу не требуется доступ к объекту, на котором настроен пользовательский сенсор мониторинга или особых прав пользователя.

В параметре объекта будет сохранён идентификатор файла.

Пример структуры сохраненного параметра

```
{
    "type_id": 8,
    "id": "6650b847742388c0efe0b687",
    "name": "file_1",
    "value": {
        "name": "test.png",
        "type": "file",
        "id": "6650bc84742388c0efe0b68b"
    }
}
```

2.2.2.4. Использование утилит операционной системы

Если в операционной системе уже установлены утилиты, которые можно использовать для сбора данных и мониторинга, то необязательно "оборачивать" их запуск в скрипт – достаточно в директории с пользовательскими сенсорами создать ссылку на исполняемый файл и добавить файл конфигурации.

Рассмотрим в качестве примера процесс добавления программы **cURL**.

1. Перейти в директорию с пользовательскими проверками:

```
$ cd /opt/saymon-custom-monitoring
```

2. Узнать путь до исполняемого файла программы **cURL**:

```
$ which curl

> /usr/bin/curl
```

3. Создать ссылку с именем **cURL** и путём до исполняемого файла /usr/bin/curl:

```
$ ln -s /usr/bin/curl cURL
```

4. Создать конфигурационный файл для проверки:

nano cURL.conf

```
{
    "title": "cURL reuse",
     "args": [
        {
            "name": "URL to cURL",
            "description": "e.g. https://google.com",
            "required": true
        }
     ]
}
```

После обновления страницы в web-интерфейсе в выпадающем списке проверок появится новый сенсор с именем **cURL reuse**. Данная проверка имеет всего один обязательный для заполнения аргумент.

2.2.3. 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.

2.2.3.1. Список идентификаторов стандартных сенсоров

В системе есть следующие стандартные сенсоры:

- **binaryProtocol** – monitoring:sensors/binary-protocol.html,
- **configFile** – monitoring:sensors/configuration-file-directory.html,
- **databaseQuery** – monitoring:sensors/database-query.html,
- **exec** – monitoring:sensors/program-script-execution.html,
- **external** – внешняя проверка (записанные через REST API данные),
- **ftp** – monitoring:sensors/ftp-probe.html,
- **httpRequest** – monitoring:sensors/http-request.html,
- **jmx** – monitoring:sensors/jmx-probe.html,
- **localNetworkPort** – monitoring:sensors/local-port.html,
- **mqtt** – monitoring:sensors/mqtt-probe.html,
- **ping** – monitoring:sensors/ping-probe.html,
- **processInfo** – monitoring:sensors/process-by-name.html,
- **prometheus** – monitoring:sensors/prometheus.html,
- **remoteNetworkPort** – monitoring:sensors/remote-port.html,
- **snmpGet** – monitoring:sensors/snmp-get.html,
- **snmpTrap** – monitoring:sensors/snmp-trap.html,
- **sslCheck** – monitoring:sensors/ssl.html,

ssh – monitoring:sensors/ssh.html,

- **wmi** – monitoring:sensors/wmi-probe.html.
- **winrm** – monitoring:sensors/winrm.html.

2.2.4. Выполнение скриптов при создании и удалении объектов

Система может выполнять скрипты при создании или удалении объектов того или иного класса. Этим скриптам можно передавать 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.

2.2.4.1. 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
```

2.2.4.2. Параметры скриптов

Параметры скрипта задаются в JSON-файле, именуемом как **имя_исполняемого_файла.conf**. Например, для скрипта **myScript.sh** файл настроек параметров должен называться **myScript.sh.conf**.

Конфигурационный файл должен располагаться в одном каталоге с исполняемым файлом.

CAUTION

Скрипты без конфигурационного файла игнорируются.

Файл настроек параметров может содержать следующий набор полей:

Поле	Описание
title	Имя скрипта, отображаемое в web-интерфейсе в списках скриптов. Опциональное поле. По умолчанию в качестве имени отображается идентификатор скрипта, генерируемый сервером из пути и имени скрипта: • путь к файлу: *<путь до папки со скриптами>***/cpu/script_1.sh** • отображаемое имя: **cpu - script_1**
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.

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). **TIP** 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.**required**	Признак обязательности заполнения значения аргумента. • **true** - значение обязательно, • **false** - значение опционально. Опциональное поле. По умолчанию - **false**.

Поле	Описание
args.type	Тип значения аргумента. Опциональное поле. По умолчанию - **text**.
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.

2.2.4.3. 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",
```

2.2.5. Модуль аналитики

Модуль аналитики – программный продукт, который позволяет обнаруживать всплески значений измеряемых метрик и прогнозировать значения метрик через указанный временной интервал.

2.2.5.1. Установка модуля аналитики

По умолчанию модуль аналитики, используемый для прогнозирования и определения всплесков, не включен в ISO-образ системы Центральный Пульт. 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 системы Центральный Пульт.
2. Запустить скрипт командой

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

TIP

x.y.z - версия модуля аналитики. Доступные версии модуля аналитики:

- 1.0.0
- 1.0.1
- 1.0.2

NOTE

Проверить, что контейнер **saymon-analytics** запустился и работает, можно с помощью команды

```
$ docker ps -a
```

2.2.5.2. Включение модуля аналитики

Чтобы включить установленный модуль аналитики, потребуется внести изменения в конфигурацию системы Центральный Пульт:

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 системы Центральный Пульт:

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

2.2.5.3. Использование модуля аналитики

Модуль аналитики может использоваться для изменения состояния объекта при определении всплеска или на основе прогноза значения.

Подробную инструкцию по настройке условий перехода состояний с помощью модуля аналитики можно найти на странице Определение всплесков и прогнозирование.

2.2.5.4. Мониторинг работы модуля аналитики

Чтобы производить мониторинг работы модуля аналитики, можно использовать журнал контейнера **saymon-analytics**. Для этого необходимо:

1. Подключиться к контейнеру командой

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

2. Открыть файл журнала модуля командой

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

В журнале 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.

2.2.5.5. Производительность модуля аналитики

Аналитика по каждому из объектов выполняется в отдельном потоке вычислений с использованием одного логического ядра системы. Все значения метрик по одному и тому же объекту обрабатываются последовательно, что позволяет обеспечивать последовательное обновление прогнозной модели в соответствии с порядком поступления данных.

Как правило, производительность современных процессоров позволяет обрабатывать каждое поступающее значение в реальном времени с максимальной частотой в одно значение метрики в секунду. Однако на перегретых или перегруженных машинах производительность ядер может падать и

приводить к накоплению задач для модуля аналитики и увеличению времени реакции системы на новые данные.

Для увеличения быстродействия системы в задачах с высокой частотой получения метрик рекомендуется выделять по одному логическому ядру на каждый объект с аналитикой. При наличии дополнительных ядер система также может использовать их для препроцессинга задач аналитики, поскольку блокировка на прогнозной модели занимает лишь часть от общего времени обработки задачи по аналитике.

2.2.6. Подключение и настройка Elasticsearch

Elasticsearch – система, которая позволяет хранить данные различного формата и осуществлять по ним полнотекстовый поиск. Эта статья содержит описание процесса настройки Elasticsearch для системы Центральный Пульт.

В предлагаемом решении для сборки и фильтрации данных используется Logstash, для настройки и визуализации данных применяется Kibana. Упомянутые три инструмента образуют так называемый стек ELK (Elasticsearch, Logstash, Kibana).

WARNING

Все описанные ниже инструкции необходимо выполнить на том сервере, где установлен Центральный Пульт.

2.2.6.1. Настройка и запуск ELK

WARNING

Если Docker Compose не установлен на вашей системе, необходимо установить его по инструкции.

Для запуска ELK необходимо выполнить следующие шаги:

1. Скачать и распаковать архив **elk.tar.gz**.
2. Перейти в директорию `elk` и выполнить следующую команду:

```
$ sudo chmod 777 elasticsearch/data/
```

3. Запустить ELK:

```
$ sudo docker-compose up -d
```

После запуска Logstash автоматически создаёт индекс для хранения данных с именем `my_index`. Далее задаются настройки для индекса.

Чтобы настроить индекс, необходимо открыть консоль в Kibana (`https://YOUR_HOST_IP:5601/app/kibana#/dev_tools/console?_g=()`) и ввести следующую команду:

```
PUT _template/my_index
{
    "index_patterns": [
        "my_index*"
    ],
    "settings": {
        "index": {
            "lifecycle": {
                "name": "my_index-policy",
                "rollover_alias": "my_index"
            }
        }
    },
    "mappings": {
        "dynamic": false,
        "properties": {
            "payload": {
                "dynamic": true,
                "type": "object"
            },
            "entityId": {
                "norms": false,
                "type": "text",
                "fields": {
                    "keyword": {
                        "ignore_above": 256,
                        "type": "keyword"
                    }
                }
            },
            "timestamp": {
                "format": "epoch_millis",
                "type": "date"
            }
        }
    },
    "aliases": {}
}
```

Если запрос выполнен успешно, то в правой части экрана будет следующий ответ:

```
{
    "acknowledged" : true
}
```

2.2.6.2. Настройка системы Центральный Пульт для работы с ELK

Для настройки ELK необходимо выполнить следующие шаги:

1. В конфигурационный файл сервера `/etc/saymon/saymon-server.conf` добавить следующие настройки:

```
"elasticsearch": {
    "node": "http://127.0.0.1:9200",
    "index_pattern": "my_index*"
}
```

2. Перезапустить сервер системы Центральный Пульт:

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

2.2.6.3. Получение данных из Elasticsearch

На текущий момент пользователи системы Центральный Пульт могут получить данные из Elasticsearch только через REST API (см. метод Получить историю данных мониторинга).

2.2.7. Передача аварийных сообщений в сторонние сервисы

Центральный Пульт может запускать заданный скрипт при возникновении аварии. Этот функционал можно использовать для передачи данных об аварии 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.

2.2.8. Самоконтроль сервера

Центральный Пульт поддерживает планирование выполнения проверки состояния памяти сервера (самоконтроль). В ходе этой проверки в указанный в конфигурации объект будут отправляться данные самоконтроля.

Самоконтроль настраивается при помощи следующих параметров серверной конфигурации:

- `server.self_object_id` — ID объекта, используемого для самоконтроля сервера,
- `server.self_monitoring_period` — Период между процессами самоконтроля сервера.

Процесс самоконтроля возвращает следующие данные:

- Информация об утечках памяти,
- Общее состояние памяти.

После выполнения процесса самоконтроля данные сохраняются в объекте, ID которого указано в параметре `server.self_object_id`. Также сервер сохраняет дамп памяти в следующий файл:

```
/tmp/saymon-rest-server-worker-<ID-объекта-самоконтроля>-<дата-и-время-дампа>.heapsnapshot
```

WARNING

Если объект с ID, указанным в `server.self_object_id` не существует, то проверка не выполняется.

Результат самоконтроля имеет следующие поля:

Поле	Описание
entityType	Числовой тип объекта системы, который использовался для процесса самоконтроля.
entityId	ID объекта системы, который использовался для процесса самоконтроля.
period	Период между запуском самоконтроля.
timestamp	Время запуска самоконтроля.
payload	Данные самоконтроля.
payload.eventLoopLagMs	Задержка работы основного потока сервера в миллисекундах.
payload.memory	Данные о памяти сервера.
payload.memory.leak	Есть ли утечка памяти на сервере?
payload.memory.leakInfo	Информация об утечке памяти.
payload.memory.usage	Информация об использовании памяти процессом сервера.
payload.memory.stats	Статистики использования памяти.

2.2.8.1. Использование памяти сервером

Пример статистики использования памяти

```
{
  rss: 23851008,
  heapTotal: 4907008,
  heapUsed: 2905912,
  external: 951886,
  arrayBuffers: 17574
}
```

2.2.8.2. Статистики использования памяти

Пример статистики использования памяти

```
{
  gcScavengeCount: 1,
  gcScavengeTime: 1100880, // ns
  gcMarkSweepCompactCount: 2,
  gcMarkSweepCompactTime: 21157231, // ns
  gcIncrementalMarkingCount: 0,
  gcIncrementalMarkingTime: 0, //ns
  gcProcessWeakCallbacksCount: 0,
  gcProcessWeakCallbacksTime: 0, // ns
  total_heap_size: 16097280, // bytes
  total_heap_size_executable: 3670016, // bytes
  total_physical_size: 10741880, // bytes
  total_available_size: 1487689928, // bytes
  used_heap_size: 5691584, // bytes
  heap_size_limit: 1501560832, // bytes
  malloced_memory: 8192,
  peak_malloced_memory: 1185464,
  gc_time: 4587251 // ns
}
```

2.2.9. Запуск сервера через PM2

В этой статье описаны шаги для запуска программного комплекса через менеджер процессов PM2.

2.2.9.1. Установка PM2

На хосте с программным комплексом необходимо установить PM2 командой:

```
$ sudo npm install -g pm2
```

2.2.9.2. Конфигурация акторов

Для всех акторов необходимо выставить режим **in-memory** в файле `/etc/saymon/actors.json`, например:

```
"RestServerWorkerActor": {
    "mode": "in-memory",
    "onCrash": "respawn"
}
```

2.2.9.3. Отключение сервиса

Если ранее на хосте программный комплекс запускался как сервис, то данный сервис необходимо остановить и отключить.

Команда для остановки сервиса:

```
$ sudo service saymon-server stop
```

Команда для отключения сервиса для systemctl:

```
$ sudo systemctl disable saymon-server
```

Команда для отключения сервиса для init:

```
$ rm /etc/init/{saymon-server.conf,saymon-server-binary.conf}
```

2.2.9.4. Запуск через PM2

Запуск программного комплекса через PM2 осуществляется командой:

```
$ sudo -Hu saymon pm2 start /usr/local/saymon/server/ecosystem.config.js
```

Чтобы сервисы стартовали автоматически при запуске операционной системы необходимо выполнить команду:

```
$ sudo -Hu saymon pm2 startup
```

2.2.9.5. Настройка журнала

Логи по каждому актору пишутся в папку `/home/saymon/.pm2/logs`.

Для настройки ротации журнала необходимо создать файл с настройками `/etc/logrotate.d/pm2-saymon` и добавить в него следующее содержимое:

```
/home/saymon/.pm2/pm2.log /home/saymon/.pm2/logs/*.log {
  maxsize 100M
  daily
  missingok
  rotate 3
  compress
  notifempty
  copytruncate
}
```

Следующая команда очищает файлы журнала:

```
$ sudo -Hu saymon pm2 flush
```

2.2.9.6. Работа с акторами

О статусе акторов можно узнать, выполнив команду

```
$ sudo -Hu saymon pm2 status
```

Мониторинг работы акторов в режиме реального времени:

```
$ sudo -Hu saymon pm2 monit
```

Для остановки/запуска/перезапуска конкретного или всех акторов необходимо использовать следующие команды:

```
$ sudo -Hu saymon pm2 start all - запуск всех акторов
$ sudo -Hu saymon pm2 restart all - перезапуск всех акторов
$ sudo -Hu saymon pm2 stop all - остановка всех акторов

$ sudo -Hu saymon pm2 start 4 5 - запуск акторов с номерами 4 и 5
$ sudo -Hu saymon pm2 restart 4 5 - перезапуск акторов с номерами 4 и 5
$ sudo -Hu saymon pm2 stop 4 5 - остановка акторов с номерами 4 и 5
```

Для масштабирования акторов следует использовать команду:

```
$ sudo -Hu saymon pm2 scale saymon-rest-server 4
```

где:

- saymon-rest-server – имя актора;
- 4 – желаемое количество акторов.

При этом перезапуск программного комплекса не требуется.

2.2.10. Включение самостоятельной регистрации пользователей

По умолчанию Центральный Пульт позволяет добавлять новых пользователей только администраторам или другим пользователям с правами на создание пользователей через меню Пользователи, Keycloak или LDAP.

Также, в системе есть функционал самостоятельной регистрации пользователей через web-interface:register-user.html.

WARNING

Параметр `server.user.auth_enabled` также включает возможность удалить собственную учётную запись.

Для включения этой функции необходимо выполнить следующие действия:

1. В подразделе **User** раздела **Server** конфигурационного файла сервера `/etc/saymon/saymon-server.conf` установить значение параметра `auth_enabled` равным `true`:

```
"server": {
    "user": {
        "auth_enabled" : true
    }
}
```

2. В том же разделе настроить Email-сервер для отправки пользователю письма с подтверждением регистрации.
3. В конфигурационном файле клиента `/etc/saymon/saymon-client.yaml` добавить параметр `authEnabled` и установить его значение равным `true`:

```
authEnabled: true
```

4. Перезапустить Центральный Пульт:

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

WARNING

Параметр `server.user.auth_enabled` также включает возможность удалить собственную учётную запись.

2.2.11. Keycloak

Вы можете использовать внешний сервер Keycloak для авторизации пользователей.

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.

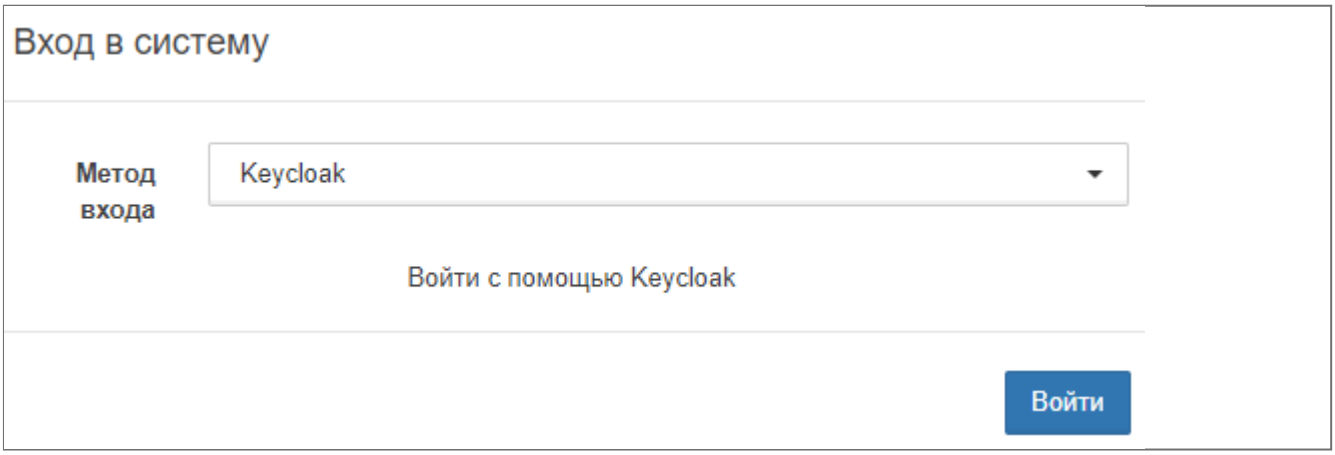


Рис. 3. Окно выбора метода входа

2.2.11.1. 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. Также, на вкладке Общие Keycloak-пользователям недоступен параметр **Активен до**.

2.2.11.2. 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**.

2.2.12. LDAP

Центральный Пульт поддерживает аутентификацию пользователей с помощью сервера службы каталогов по протоколу LDAP (Active Directory, OpenLDAP, и т.д.).

2.2.12.1. Настройка LDAP

По умолчанию, аутентификация 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 системе Центральный Пульт.
- `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 системе Центральный Пульт, до LDAP-групп при совпадении имен групп.

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

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

2.2.12.2. Аутентификация в системе Центральный Пульт 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 .

Также, на вкладке Общие LDAP-пользователям недоступен параметр **Активен до**.

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 системе Центральный Пульт, 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 системе Центральный Пульт, 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 системе Центральный Пульт, 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 системе Центральный Пульт - 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.

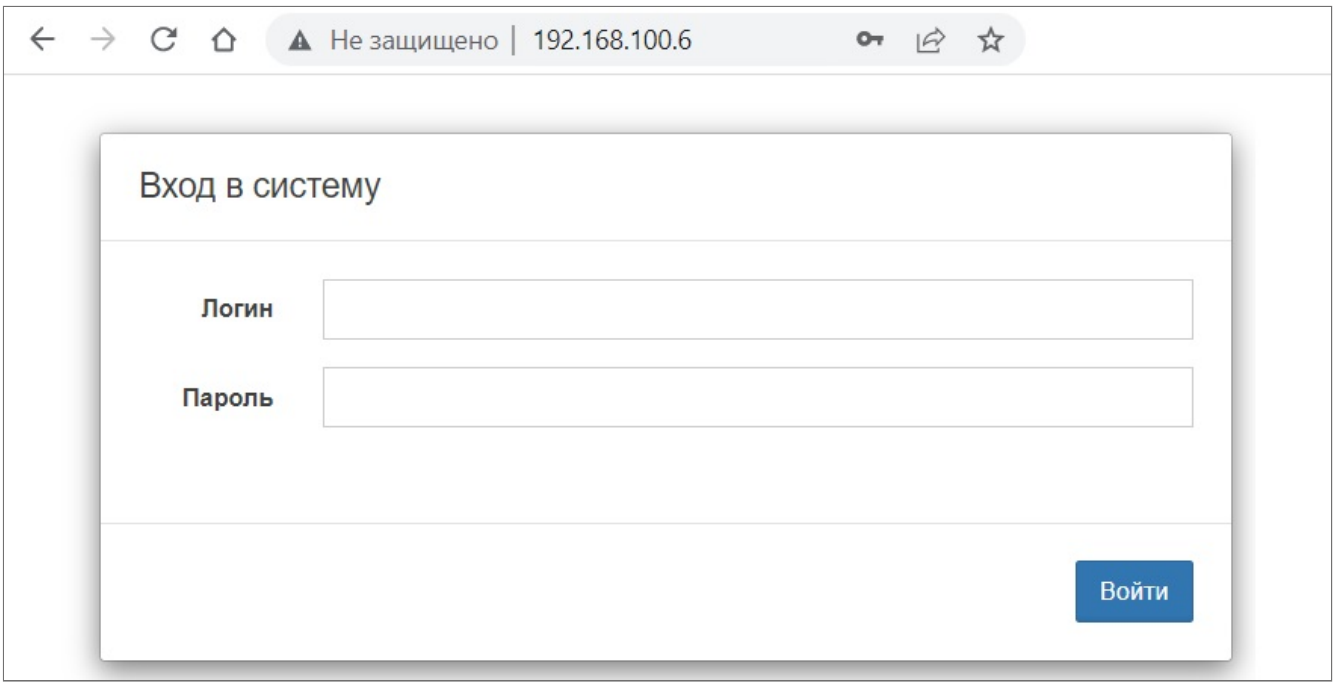
TIP

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. Подключение к серверу

3.1. Подключение к системе через Web-интерфейс



Для подключения к серверу через web-интерфейс необходимо выполнить следующие действия:

1. В адресной строке браузера ввести адрес web-сервера.

TIP

Адрес web-сервера отображается в диалоговом окне после завершения установки серверной части системы Центральный Пульт (подробнее см. server-installation:virtualbox.html).

2. В открывшемся диалоговом окне ввести ваш логин и пароль.
3. Нажать кнопку Войти или `Enter`.

Стандартный пароль от аккаунта администратора:

- **Логин** – `admin`;
- **Пароль** – `saymon`.

NOTE

Выше указаны логин и пароль от учётной записи с правами администратора. В целях безопасности при первом входе через учётную запись `admin` система потребует смены пароля.

Также, на этом экране можно выбрать язык системы с помощью меню в правом верхнем углу. Это меню активно только если в конфигурации Web-приложения определено несколько языков.

3.2. Подключение к системе через REST API

TIP

Подробную информацию о работе с системой с помощью REST API можно прочитать в документации REST API.

Для отправки большинства запросов к серверу системы Центральный Пульт нужно авторизоваться. Система поддерживает три способа авторизации пользователя:

- Авторизация с помощью аккаунта системы;
- Авторизация с помощью токена сессии;
- Авторизация с помощью токена API.

3.3. Подключение к серверу по SSH

Для подключения к серверу по SSH необходимо выполнить следующие действия:

1. В командной строке ввести:

```
$ssh <имя-пользователя>@<IP-адрес web-сервера>
```

2. В открывшемся диалоговом окне ввести пароль пользователя.

Логин и пароль пользователя системы Центральный Пульт не подойдёт для авторизации через SSH. Для получения доступа, обратитесь к администратору.

NOTE

Стандартные данные для авторизации администратора:

- **Логин** - `saymon`;
- **Пароль** - `saymon`.

4. Агент SAYMON

В этом разделе описан процесс установки агента на различные операционные системы и процесс обновления установленных агентов, описание конфигурационных файлов агента и его логирования, а также настройка автообнаружения объектов мониторинга.

4.1. Установка агента

Агента можно установить на различные операционные системы, поддерживающие Java SE 17 или позднее. Агент также может быть установлен на контроллер автоматизации Wiren Board на базе Linux.

Если по каким-то причинам установка агента нежелательна или невозможна, можно настроить безагентный мониторинг системы.

NOTE

Подробнее о типах мониторинга и общей архитектуре процесса мониторинга в системе Центральный Пульт написано в статье monitoring:architecture.html.

4.1.1. Требования для установки агента системы

Агент может быть установлен на выделенных аппаратных или виртуальных мощностях.

Минимальные аппаратные требования для работы агентов системы:

- Операционная система с поддержкой Java 17;
- Процессор — 1 ядро, частота 2 ГГц;
- Оперативная память — 1 Гб;
- Жёсткий диск — 2 Гб.

4.1.2. Windows

4.1.2.1. 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**.

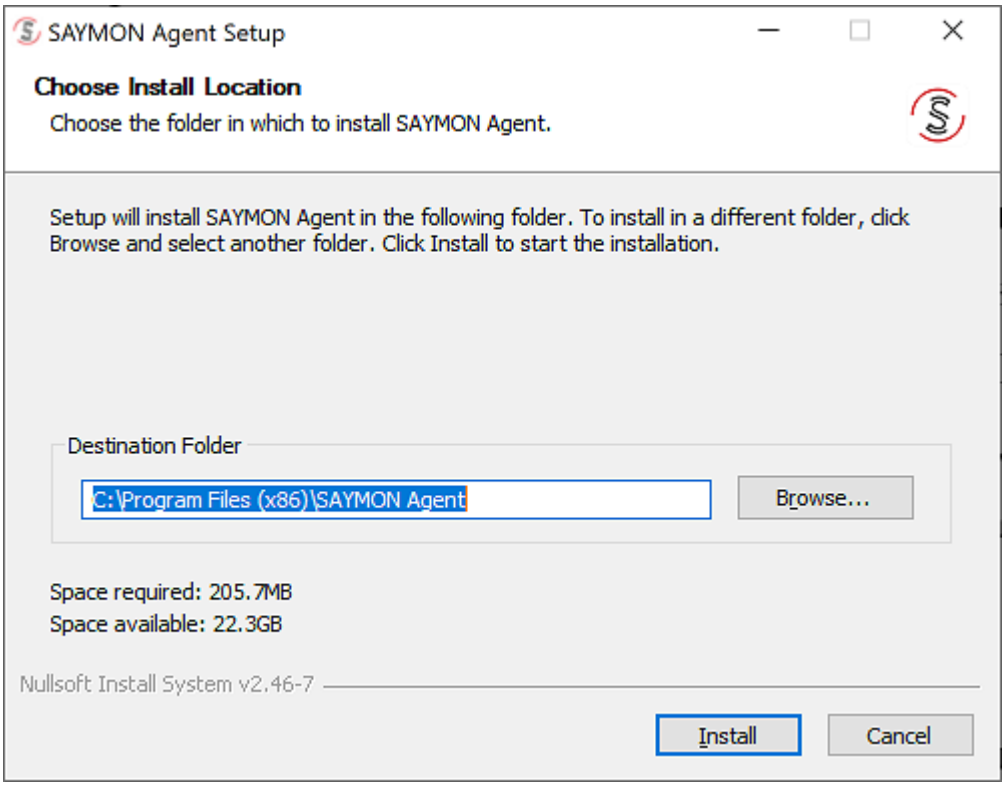


Рис. 4. Выбор каталога установки

4. После завершения установки нажать кнопку **Next**.

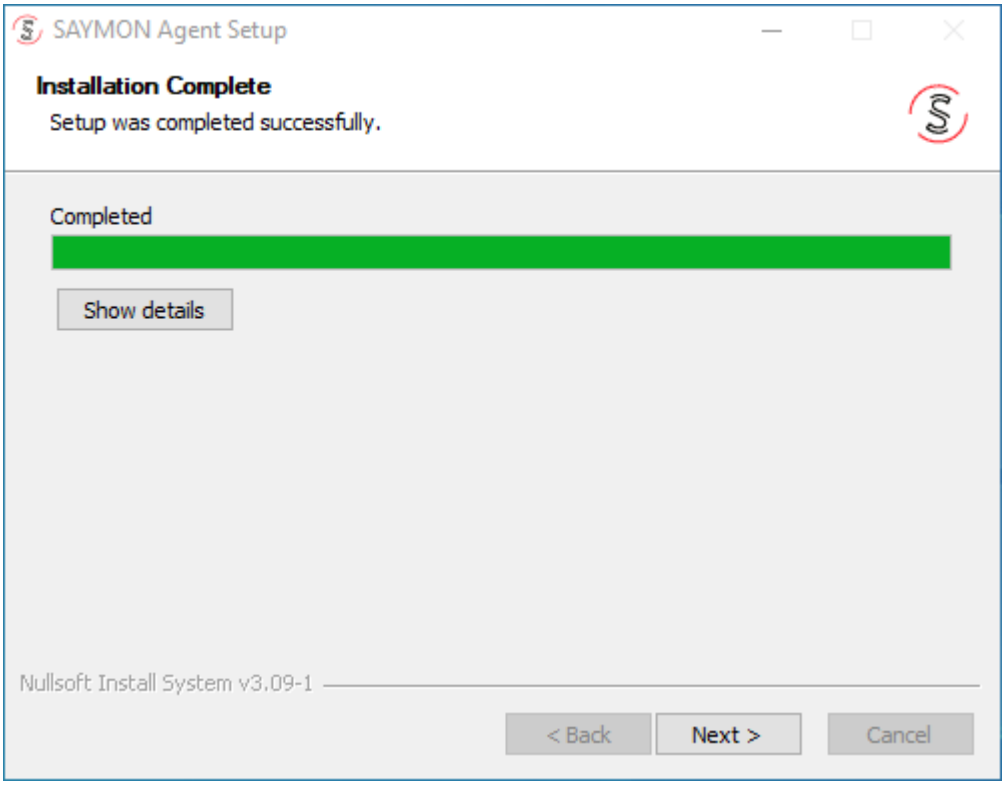


Рис. 5. Завершение установки

5. В открывшемся окне можно выполнить первичную настройку агента. Пункты меню соответствуют следующим параметрам Конфигурационного файла агента:
   - **SAYMON Server address** – `server.host`
   - **SAYMON Agent ID** – `agent.id`
   - **Batch / PowerShell scripting support** – `agent.scriptsEnabled`

   NOTE

   По умолчанию, поле **SAYMON Agent ID** имеет значение 0. Если оставить это значение, то агент попытается зарегистрироваться на сервере автоматически.

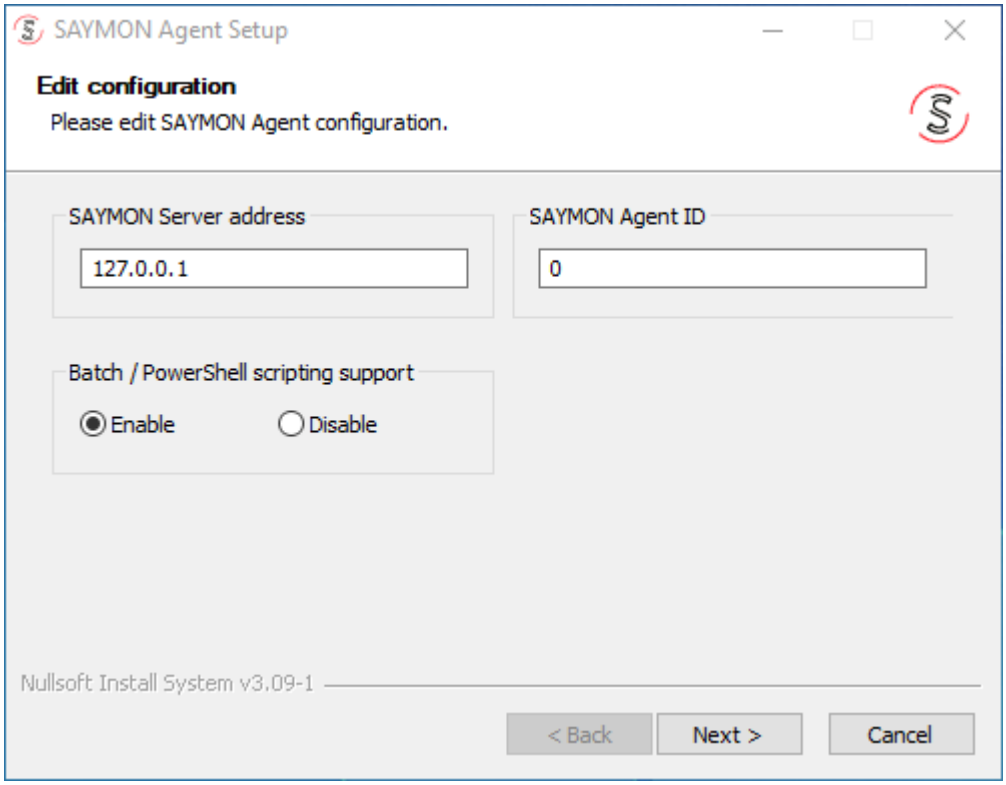
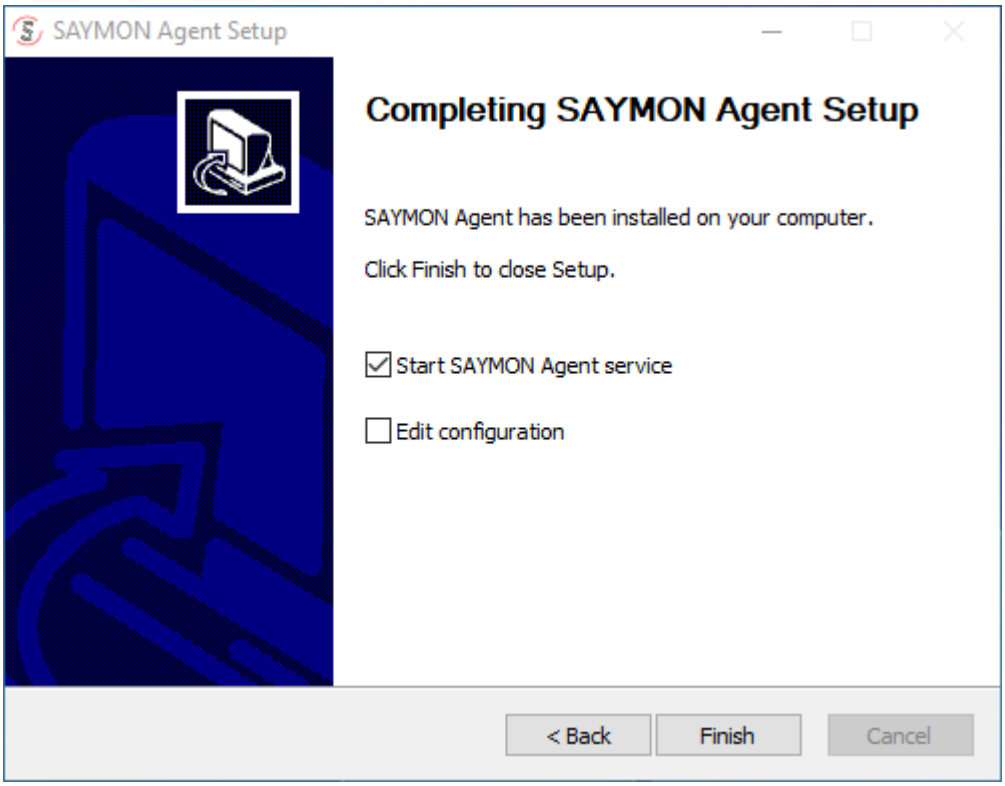


Рис. 6. Первичная настройка агента

6. После настройки агента, нажать кнопку **Next**.

7. Установка агента завершена. В этом окне можно отметить действия, которые нужно совершить после установки. **Start SAYMON Agent Service** запускает установленного агента, а **Edit configuration** открывает (Конфигурационный файл агента). Также, файл конфигурации можно открыть вручную. Файл `agent.properties` находится в папке `conf` каталога установки агента.



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.

Чтобы инициировать процедуру получения `agentId` повторно (например, если нужно подключить агента к другому серверу), достаточно удалить файл `registration.json` и перезапустить службу агента.

4.1.2.2. Log-файлы

Во время работы агент создает log-файлы. Эти файлы хранятся в папке `log` каталога установки агента.

4.1.2.3. 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.

4.1.3. Linux

Агент SAYMON для Linux поставляется в виде установочного скрипта и пакетов для пакетных менеджеров Ubuntu/Debian и Red Hat/Fedora. Оба способа установки не требуют интернета.

В примерах к этой инструкции пакеты и установочные скрипты скачиваются отдельной командой с помощью `wget`. Чтобы установить агент на хост без доступа к интернету нужно скачать установщики агента заранее и вручную перенести его на нужный хост.

Оба способа установки поддерживают автоматическое обновление.

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.

После установки автоматически запускается сервис `saymon-agent`.

4.1.3.1. Установка с помощью скрипта

Для установки агента с помощью установочного скрипта нужно получить скрипт для нужной версии агента и запустить его на нужном устройстве:

```
$wget https://downloads.cpult.ru/agent/v5.1.1/saymon-agent-5.1.1-linux-x64-installer.sh
$sudo sh saymon-agent-5.1.1-linux-x64-installer.sh
```

WARNING

Для каждой архитектуры системы существует свой установочный скрипт. При установке и обновлении скриптом неподходящей архитектуры агент будет перезаписан, но сервис не запустится. Исправить некорректную установку можно обновив агент скриптом для нужной архитектуры.

При запуске скрипта пользователю будет показано интерактивное меню, где он может выбрать способ конфигурации агента при установке или отменить установку:

```
Unpacking the distribution archive... OK
Please select how you want to configure SAYMON Agent:
1) Manual configuration.
2) Auto configuration.
3) Quit without installing.
```

1. Ручная конфигурация;
2. Автоматическая конфигурация;
3. Завершить работу скрипта без установки агента.

Если скрипт обнаружит установленного агента в папке `/opt/saymon-agent`, то предложит его обновить. Инструкцию по установке второй копии агента на эту же систему можно найти на странице ./extra-agents-installation.html.

Ручная конфигурация

При установке агента с ручной конфигурацией скрипт предлагает пользователю ввести ID агента и адрес сервера, на котором нужно зарегистрировать агента. После этого он распаковывает архив с агентом в папку `/opt/saymon-agent`, создаёт группу `saymon` и пользователя `saymon`, включает сервис `saymon-agent` в `systemd`.

Вывод в консоль во время установки с ручной конфигурацией

```
Unpacking the distribution archive... OK
Please select how you want to configure SAYMON Agent:
1) Manual configuration.
2) Auto configuration.
3) Quit without installing.
Your choice: 1
Please enter agent ID (leave empty or enter 0 for agent ID auto registration):
5e21b752308c3c66d64e072c
SAYMON Agent ID set to '5e21b752308c3c66d64e072c'.
Please enter SAYMON Server address (IP or hostname): 192.168.1.45
SAYMON Server address set to '192.168.1.45'.
Created 'saymon' user group with GID 11813.
Created 'saymon' user with UID 11813.
SAYMON Agent installed to /opt/saymon-agent.
Systemd detected.
SAYMON Agent systemd service installed and enabled.
Start SAYMON Agent service.
SAYMON Agent installation completed.
```

Автоматическая конфигурация

При установке агента с автоматической конфигурацией скрипт распаковывает архив с агентом в папку `/opt/saymon-agent`, создаёт группу `saymon` и пользователя `saymon`, включает сервис `saymon-agent` в `systemd`, а также создаёт стандартный файл конфигурации, который включает автоматическую регистрацию агента на сервере, расположенном по адресу `127.0.0.1`. Адрес сервера можно настроить после установки агента.

```
Unpacking the distribution archive... OK
Please select how you want to configure SAYMON Agent:
1) Manual configuration.
2) Auto configuration.
3) Quit without installing.
Your choice: 2
SAYMON Agent will be installed with agent ID auto registration enabled and default SAYMON
server address.
Please edit /opt/saymon-agent/conf/agent.properties if you want to change these settings
later.
SAYMON Agent ID set to '0'.
Created 'saymon' user group with GID 11813.
Created 'saymon' user with UID 11813.
SAYMON Agent installed to /opt/saymon-agent.
Systemd detected.
SAYMON Agent systemd service installed and enabled.
Start SAYMON Agent service.
SAYMON Agent installation completed.
```

При выборе автоматической конфигурации будет сгенерирован следующий файл конфигурации:

```
#
# SAYMON Agent configuration file.
#

# Value 0 means SAYMON Agent will attempt to register on server automatically.
agent.id=0

#  Secure connection to SAYMON Server. Enabled by default.
server.secure=false

# SAYMON Server address.
server.host=127.0.0.1
# server.port=9092

# Network hosts discovery.
agent.discoveryEnabled=false

# Allows SAYMON Agent to execute custom shell scripts.
agent.scriptsEnabled=true

# SNMP traps receiver.
agent.snmpTrapEnabled=false
# USM username, authentication and privacy password should be 8+ characters long.
# agent.snmpTrapUsmUser=
# agent.snmpTrapUsmAuthenticationPassword=
# agent.snmpTrapUsmPrivacyPassword=
```

4.1.3.2. Установка с помощью пакетного менеджера

Для установки агента через пакетные менеджеры нужно получить пакет с версией агента для своего пакетного менеджера (`.deb` для систем Debian/Ubuntu и `.rpm` для Red Hat/Fedora) и установить его на

нужном устройстве:

deb

```
$wget https://downloads.cpult.ru/agent/v5.1.1/saymon-agent_5.1.1-cd6973ea_amd64.deb
$sudo dpkg -i saymon-agent_5.1.1-cd6973ea_amd64.deb
```

rpm

```
$wget https://downloads.cpult.ru/agent/v5.1.1/saymon-agent-5.1.1-cd6973ea.x86_64.rpm
$sudo dnf install saymon-agent-5.1.1-cd6973ea.x86_64.rpm
```

При успешной установке пакетов, в консоль выведутся следующие сообщения:

deb

```
Selecting previously unselected package saymon-agent.
(Reading database ... 37334 files and directories currently installed.)
Preparing to unpack saymon-agent_5.1.1-cd6973ea_amd64.deb ...
passwd: password changed.
Unpacking saymon-agent (5.1.1-cd6973ea) ...
Setting up saymon-agent (5.1.1-cd6973ea) ...
You can configure /opt/saymon-agent/conf/agent.properties
Backup directory /opt/saymon-agent.backup found. Also you can restore the
configuration from there.
And restart the service
sudo systemctl restart saymon-agent
Start the service
Created symlink /etc/systemd/system/multi-user.target.wants/saymon-agent.service →
/etc/systemd/system/saymon-agent.service.
```

rpm

```
Last metadata expiration check: 0:13:14 ago on Tue 06 Aug 2024 03:36:47 PM MSK.
Dependencies resolved.
================================================================================
================================================================================
==
 Package                                 Architecture                        Version
Repository                               Size
================================================================================
================================================================================
==
Installing:
 saymon-agent                              x86_64                              5.1.1-
cd6973ea                                @commandline                             90
M

Transaction Summary
================================================================================
================================================================================
==
Install  1 Package

Total size: 90 M
Installed size: 186 M
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
1/1
  Running scriptlet: saymon-agent-5.1.1-cd6973ea.x86_64
1/1
passwd: password changed.

  Installing       : saymon-agent-5.1.1-cd6973ea.x86_64
1/1
  Running scriptlet: saymon-agent-5.1.1-cd6973ea.x86_64
1/1
You can configure /opt/saymon-agent/conf/agent.properties
Backup directory /opt/saymon-agent.backup found. Also you can restore the
configuration from there.
And restart the service
sudo systemctl restart saymon-agent
Start the service
Created symlink /etc/systemd/system/multi-user.target.wants/saymon-agent.service →
/etc/systemd/system/saymon-agent.service.


Installed:
  saymon-agent-5.1.1-cd6973ea.x86_64

Complete!
```

4.1.3.3. Регистрация агента

При установке агента на хост, происходит регистрация агента. 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** (`/opt/saymon-agent/storage`), в котором указан данный идентификатор.

Чтобы инициировать процедуру получения **agentId** повторно (например, если нужно подключить агента к другому серверу), нужно удалить файл **registration.json** и перезапустить службу агента.

4.1.3.4. Журналирование

По умолчанию файлы журнала агента создаются в директории `/var/log/saymon` .

При необходимости можно включить централизованное журналирование агента через SYSLOG в файле `/opt/saymon-agent/conf/logback-upstart.xml` .

4.1.3.5. Работа службы агента

Узнать состояние, запустить, перезапустить и остановить службу агента можно с помощью следующих команд:

```
$ sudo service saymon-agent status
$ sudo service saymon-agent start
$ sudo service saymon-agent restart
$ sudo service saymon-agent stop
```

4.1.3.6. 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).

4.1.4. macOS

4.1.4.1. Установка

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
```

4.1.4.2. Место установки

По умолчанию агент устанавливается в следующем каталоге: `/opt/saymon-agent`

4.1.4.3. Log-файлы

При необходимости можно включить централизованное журналирование агента через SYSLOG в файле `saymon-agent/conf/logback.xml`.

Файлы журнала агента записываются в директорию `/opt/saymon-agent/log`.

4.1.5. Wiren Board

Wiren Board — 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.

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

4.1.5.1. Установка

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
```

4.1.5.2. Конфигурация агента

При необходимости можно отредактировать конфигурационный файл агента `/opt/saymon-agent/conf/agent.properties`.

4.1.5.3. Место установки

По умолчанию агент устанавливается в каталоге `/opt/saymon-agent`.

4.1.5.4. Файлы журнала

Файлы журнала агента записываются в директорию `/var/log/saymon`.

Журналирование отключается в файле `/opt/saymon-agent/conf/logback-upstart.xml`:

Журналирование включено

```
<root level="INFO">
<appender-ref ref="FILE-INFO"/>
<!--<appender-ref ref="FILE-DEBUG"/>-->
<!--<appender-ref ref="SYSLOG"/>-->
</root>
```

Журналирование отключено

```
<root level="INFO">
<!--<appender-ref ref="FILE-INFO"/>-->
<!--<appender-ref ref="FILE-DEBUG"/>-->
<!--<appender-ref ref="SYSLOG"/>-->
</root>
```

4.1.5.5. Работа службы

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

```
$ service saymon-agent status
$ service saymon-agent start
$ service saymon-agent restart
$ service saymon-agent stop
```

4.1.6. Автоинсталляция

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

В ходе процедуры автоинсталляции, система Центральный Пульт генерирует команду для скачивания и установки агента, который администратор должен запустить на сервере. После выполнения команды агент устанавливается в автоматическом режиме.

NOTE

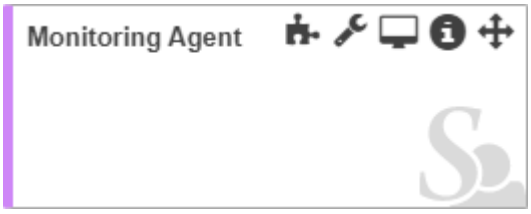
При автоинсталляции поддерживается установка только на поддерживаемые системы Linux.

WARNING

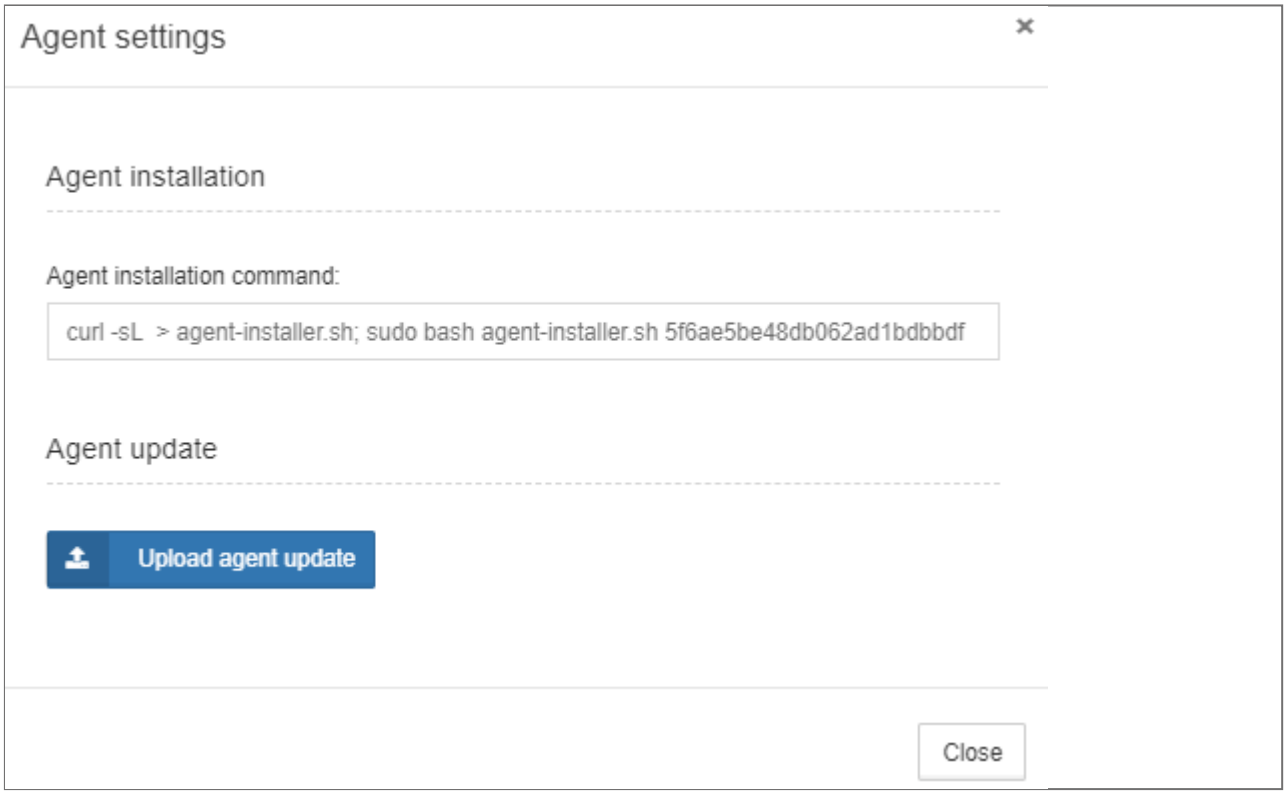
Перед выполнением процедуры автоинсталляции, задайте ссылку на скачивание установочного файла в параметре agentInstallerUrl конфигурация Web-приложения.

Для выполнения автоинсталляции агента, выполните следующие шаги:

1. Создать объект класса `Saymon Agent`.
2. Навести курсор на созданный объект и нажать на появившуюся иконку 🔧:



3. В появившемся окне скопировать текст из поля `Команда для установки агента`:



WARNING

По умолчанию ссылка на скачивание установщика агента не задана. Перед выполнением процедуры автоинсталляции, задайте ссылку на установочный скрипт в параметре agentInstallerUrl конфигурация Web-приложения.

4. Выполнить данную команду на необходимом сервере. После выполнения команды на сервере агент установится в автоматическом режиме.

NOTE

Последние 24 символа команды – уникальный ID агента. ID агента совпадает с ID соответствующего объекта класса SAYMON Agent на сервере. Также, ID агента можно найти в файле `/opt/saymon-agent/storage/registration.json`.

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

NOTE

Соответствие цвета и состояния объектов указывается в списке состояний.

TIP

После установки может потребоваться дополнительная настройка установленного агента в файле конфигурации.

4.1.7. Установка дополнительных агентов

При работе с системой может возникнуть необходимость выполнять мониторинг одного хоста на разных серверах системы Центральный Пульт. Поскольку один агент может работать только с одним сервером, в таком случае нужно установить и настроить на хосте дополнительного агента для каждого дополнительного сервера.

WARNING

Дополнительные агенты и проводимые ими проверки занимают ресурсы хоста.

4.1.7.1. Linux

Создание копии агента

1. Скопировать директорию `/opt/saymon-agent`, например, в `/opt/saymon-agent-2`:

```
$ sudo cp -r /opt/saymon-agent /opt/saymon-agent-2
```

2. Поменять владельца новой директории:

```
$ sudo chown saymon:saymon /opt/saymon-agent-2
```

3. Удалить содержимое папки `/opt/saymon-agent-2/storage`:

```
$ sudo rm -rf /opt/saymon-agent-2/storage
```

4. В файле `/opt/saymon-agent-2/conf/agent.properties` задать настройки агента:

```
$ sudo nano /opt/saymon-agent-2/conf/agent.properties
```

5. В файлах настроек журналирования везде заменить `saymon-agent` на `saymon-agent-2`.

```
$ sudo nano /opt/saymon-agent-2/conf/logback.xml
$ sudo nano /opt/saymon-agent-2/conf/logback-upstart.xml
```

Копирование скрипта запуска службы

1. Скопировать файл `/etc/systemd/system/saymon-agent.service`, например, в `/etc/systemd/system/saymon-agent-2.service`:

```
$ sudo cp /etc/systemd/system/saymon-agent.service /etc/systemd/system/saymon-agent-
2.service
```

2. В новом файле `/etc/systemd/system/saymon-agent-2.service` заменить следующие строки:

```
Environment=SAYMON_AGENT_LOGBACK_CONFIGURATION_FILE=/opt/saymon-agent/conf/logback-
upstart.xml
WorkingDirectory=/opt/saymon-agent
ExecStartPre=/bin/bash -ec "/opt/saymon-agent/saymon-agent-updater.sh >
/var/log/saymon/saymon-agent-updater.out 2>&1"
ExecStart=/opt/saymon-agent/saymon-agent.sh > /var/log/saymon/saymon-agent.out 2>&1
```

на:

```
Environment=SAYMON_AGENT_LOGBACK_CONFIGURATION_FILE=/opt/saymon-agent-2/conf/logback-
upstart.xml
WorkingDirectory=/opt/saymon-agent-2
ExecStartPre=/bin/bash -ec "/opt/saymon-agent-2/saymon-agent-updater.sh >
/var/log/saymon/saymon-agent-2-updater.out 2>&1"
ExecStart=/opt/saymon-agent-2/saymon-agent.sh > /var/log/saymon/saymon-agent-2.out 2>&1
```

Запуск агента

В терминале необходимо выполнить команды активации и запуска службы нового агента:

```
$ sudo systemctl enable saymon-agent-2
$ sudo service saymon-agent-2 start
```

4.1.7.2. Windows

Создание копии агента

1. Скопировать директорию с агентом (по умолчанию `C:\Program Files (x86)\SAYMON Agent`), например, в `C:\Program Files (x86)\SAYMON Agent 2`.
2. Удалить содержимое папки `C:\Program Files (x86)\SAYMON Agent 2\storage`.
3. В файле `C:\Program Files (x86)\SAYMON Agent 2\conf\agent.properties` задать настройки агента.

Установка службы нового агента

1. В файле `C:\Program Files (x86)\SAYMON Agent 2\saymon-agent.xml` заменить строки:

```
<id>saymon-agent</id>
<name>SAYMON Agent</name>
<description>SAYMON Agent</description>
```

на

```
<id>saymon-agent-2</id>
<name>SAYMON Agent 2</name>
<description>SAYMON Agent 2</description>
```

2.

Запустить командную строку от имени администратора.

3. Перейти в директорию `C:\Program Files (x86)\SAYMON Agent 2`:

```
$ cd "C:\Program Files (x86)\SAYMON Agent 2"
```

4. Установить и запустить службу нового агента командами:

```
$ saymon-agent.exe install
$ saymon-agent.exe start
```

4.1.8. 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`.

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.
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. **NOTE** О возможностях работы с SNMP-Trap читайте в статьях monitoring:sensors/snmp-trap.html и web-interface:event-log.html.
agent.snmpTrapListenPort	Порт для получения трапов. По умолчанию – **1162**.
agent.snmpTrapReceiverThreadPoolSize	Количество одновременных потоков для получения трапов. По умолчанию – **4**.

Параметр	Описание
agent.snmpTrapV3AuthPassword	Ключ аутентификации получателя SNMPv3 трапов. Не имеет значения по умолчанию.
agent.snmpTrapV3PrivacyPassword	Ключ шифрования для получателя SNMPv3 трапов. Не имеет значения по умолчанию.
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 5.0.0 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**.
server.secure	Подключаться к брокеру Kafka через TLS. Для использования зашифрованного подключения, брокер Kafka должен быть настроен на приём зашифрованных соединений на порте, указанном в параметре `server.port` (по умолчанию – 9092). При выключении параметра `server.secure`, подключение происходит в текстовом формате. По умолчанию – **true**. **NOTE** Так как начиная с агента версии 5.0.0 шифрование включено по умолчанию, при обновлении агента необходимо настроить Kafka для работы через TLS или установить этот параметр в `false` в файле конфигурации.

4.1.8.1. 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. TIP: Если явно указаны параметры `server.host` и `server.port`, то это свойство автоматически генерируется.

WARNING

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

4.1.8.2. 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. **TIP** Этот параметр связан с параметром `server.secure`. Если параметр `server.secure` имеет значение `false`, то этот параметр может иметь значение `PLAINTEXT`. Если параметр `server.secure` выставлен в `true` или не задан, то он этот параметр может принимать любое значение кроме `PLAINTEXT`.
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.

4.1.9. 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).

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

Агент выполняет ротацию журналов раз в день, старые журналы архивируются.

NOTE

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

4.1.9.1. Базовое журналирование

Пример секции настроек базового журналирования:

```
<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 журналов.

NOTE

Полная информация о настройках доступна в документации по Logback.

4.1.9.2. 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 журналирования.

4.1.10. Автообнаружение

Центральный Пульт позволяет описать инфраструктуру любой сложности в терминах объектов и связей между ними. Эта инфраструктура затем наполняется информацией, полученной от агентов мониторинга, виджетами с текущими показателями метрик и графиками, раскрашивается в цвета состояний.

Однако создавать карту инфраструктуры вручную зачастую слишком утомительно, ведь в ней могут содержаться тысячи объектов. Для решения этой проблемы в системе Центральный Пульт предусмотрены механизмы автообнаружения объектов. С их помощью можно добавить большинство интересующих пользователя объектов на карту инфраструктуры в автоматическом режиме, без ручного "накликивания" в web-интерфейсе.

В системе Центральный Пульт есть две разновидности автообнаружения объектов.

Первая разновидность – это встроенные механизмы, которые работают без каких-либо усилий со стороны пользователя. К ним относится, например, механизм сетевого автообнаружения, который отображает на схеме все хосты, доступные в локальной сети.

Вторая разновидность – это методы в REST API, которые могут использоваться скриптами автоматизации пользователя для создания объектов. С помощью таких скриптов можно реализовать любой алгоритм автообнаружения любых разновидностей объектов инфраструктуры.

4.1.10.1. Встроенное автообнаружение

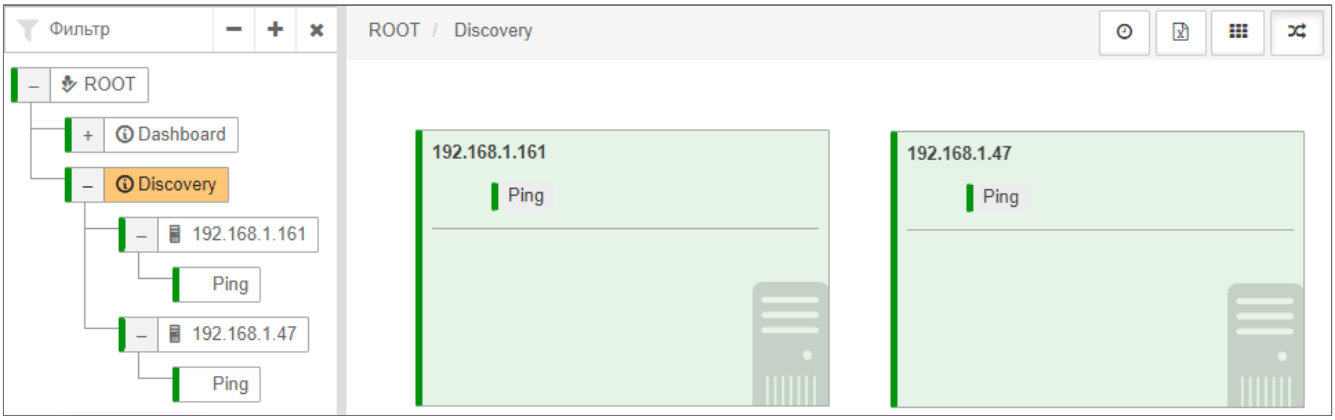
Встроенные механизмы автообнаружения реализованы внутри сервера Центральный Пульт и срабатывают автоматически при старте системы (за исключением случаев, когда они отключены администратором системы).

К встроенным механизмам автообнаружения относится механизм сетевого автообнаружения.

Сетевое автообнаружение

Агент сканирует локальную подсеть и отправляет на сервер информацию обо всех IP-адресах, доступных для TCP-соединения либо доступных по PING. Для обнаруженных хостов в системе Центральный Пульт создаются объекты класса `Host` с соответствующими адресами. Внутри каждого из этих объектов автоматически создаётся подобъект 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.



Сетевое автообнаружение включено по умолчанию и может быть выключено в конфигурационном файле агента параметром **agent.discoveryEnabled=false**:

```
agent.discoveryEnabled=false
```

По умолчанию обнаруженные объекты помещаются в корневой объект иерархии. Чтобы изменить родительский объект для автоматически обнаруживаемых объектов, необходимо указать идентификатор нового родительского объекта в параметре **server.discovery_parent_id** конфигурационного файла сервера:

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

WARNING

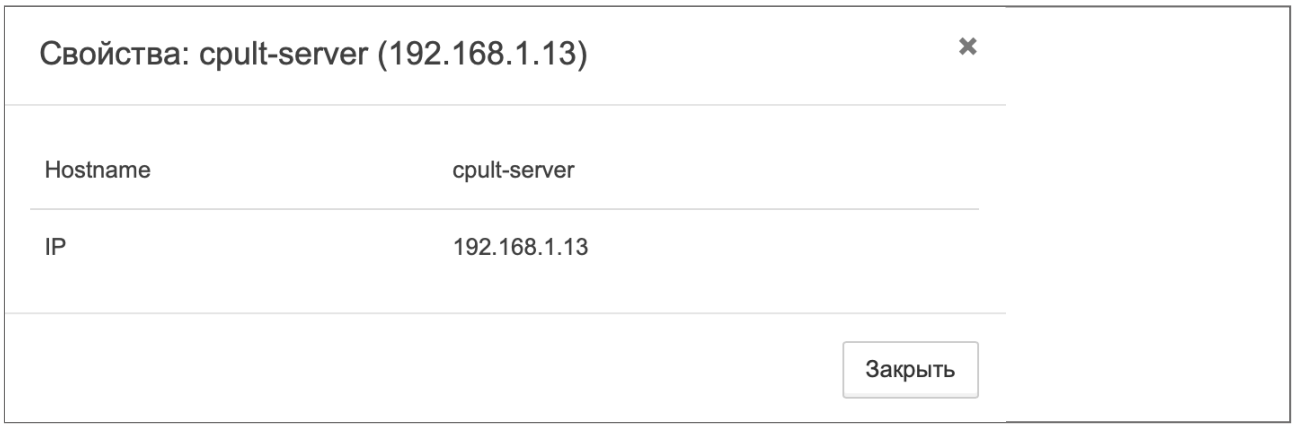
Для каждого из обнаруженных хостов может быть создан только один объект. После удаления объекта, связанного с обнаруженным хостом, новый объект для данного хоста создаваться не будет.

Авторегистрация агентов

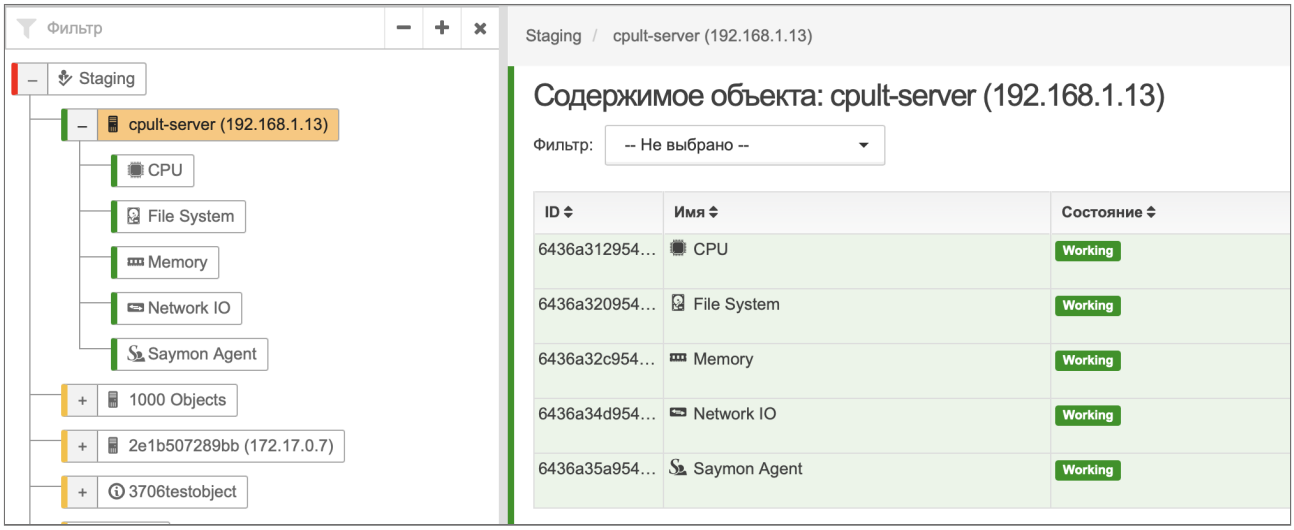
Если установить агент на хост и в настройках агента указать `agent.id=0`, то на сервере создаётся объект класса `Host`.

У созданного объекта:

- формируется имя по шаблону `имя_хоста (ip адрес хоста)`;
- добавляются свойства `Hostname` и `IP`:



- создаются следующие дочерние объекты с проверками по умолчанию:



NOTE

Для автоматического обновления имени и свойств созданного объекта в серверной конфигурации следует выставить флаг server.hostname_update.enabled в `true`.

4.1.10.2. Автообнаружение через API

Все варианты автообнаружения для всего многообразия существующих инфраструктур предусмотреть невозможно. Однако Центральный Пульт имеет открытый REST API, используя который, клиент может реализовать любой, подходящий именно ему механизм автообнаружения объектов, с помощью автоматического сценария (скрипта). REST API системы Центральный Пульт работает по протоколу HTTP, поэтому сценарий можно реализовать на любом языке или платформе, где есть HTTP-клиент.

TIP

Простейший пример – bash-скрипт с отправкой REST-запросов в Центральный Пульт через `curl`.

Добавление объектов через REST

К примеру, есть скрипт, который умеет обнаруживать определённый вид объектов в инфраструктуре (это могут быть сетевые устройства или программные сервисы). Добавить обнаруженные объекты в Центральный Пульт можно с помощью REST-метода **POST/node/api/objects.**

Пример, как это может делаться в bash-скрипте:

```
# Код для обнаружения объекта...
# ...

# Объект обнаружен, создаём его на сервере.
curl -v http://saymon.local/node/api/objects \
  -u "admin:saymon" \
  -H "Content-Type: application/json" \
  -X POST \
  -d '{"name": "Обнаруженный объект", "discovery_id": "Идентификатор_объекта"}'
```

NOTE

Больше информации о создании объектов и примеры на других языках программирования – rest-api:documentation/methods/objects/create-new-object.html

Важным параметром при обнаружении объекта является **discovery_id**. Это строка, однозначно идентифицирующая объект в инфраструктуре и позволяющая избежать его повторного создания в системе. Если в момент создания объекта на сервере уже существует объект с указанным **discovery_id**, он просто модифицируется новыми данными, а новый объект при этом не создаётся. Это позволяет запускать сценарий автообнаружения множество раз (например, по расписанию) и быть уверенным, что уже обнаруженные объекты не будут созданы повторно.

TIP

В случае с сетевыми устройствами в качестве **discovery_id** может выступать IP-адрес или MAC-адрес устройства.

4.1.11. Обновление агентов

Агенты можно обновлять с помощью тех же скриптов и программ, которые используются для установки агента. Если установочный скрипт или программа обнаружит существующую установку агента, то предложит обновить его, с возможностью сохранить конфигурацию.

TIP

Инструкцию по установке дополнительного агента на хост, без обновления существующего агента можно найти в статье installation/extra-agents-installation.html.

Примечания для обновлений конкретных версий можно найти на странице migration-notes.html.

4.1.11.1. Windows

Для обновления агента, запустите установочный файл нужной версии и следуйте инструкции по установке агента. После завершения установки, поля в окне первичной настройки агента должны быть заполнены значениями существующей конфигурации.

NOTE

До версии агента 5.0.0, установщик агента для Windows не поддерживал автоматическое определение установленных версий агента и их обновление.

Для обновления существующего агента на версию до 5.0.0 следует удалить существующего агента и установить новую версию в эту же папку. При удалении агента остаются папки `conf`, `log` и `storage`.

4.1.11.2. macOS

Агент для macOS распространяется в виде установочного скрипта. Для обновления агента нужно получить скрипт для новой версии агента и запустить его на нужном устройстве:

```
$wget https://downloads.cpult.ru/agent/v5.1.1/saymon-agent-5.1.1-macos-aarch64-installer.sh
$sh saymon-agent-5.1.1-macos-aarch64-installer.sh
```

WARNING

Для каждой архитектуры системы существует свой установочный скрипт. При установке и обновлении скриптом неподходящей архитектуры агент будет перезаписан, но сервис не запустится. Исправить некорректную установку можно обновив агент скриптом для нужной архитектуры.

```
Looks like SAYMON Agent has been already installed in /opt/saymon-agent directory.
Do you want to update application with respect to current configurations? [y/N]
y
Unpacking the distribution archive... OK
SAYMON Agent will be updated.
Copy resources from package into /opt/saymon-agent.update
Copy resources from previous installation into /opt/saymon-agent.update
'/opt/saymon-agent/storage' -> '/opt/saymon-agent.update/storage'
'/opt/saymon-agent/storage/registration.json' -> '/opt/saymon-
agent.update/storage/registration.json'
'/opt/saymon-agent/conf/agent.properties' -> '/opt/saymon-agent.update/conf/agent.properties'
'/opt/saymon-agent/conf/logback-daemon.xml' -> '/opt/saymon-agent.update/conf/logback-
daemon.xml'
'/opt/saymon-agent/conf/logback.xml' -> '/opt/saymon-agent.update/conf/logback.xml'
'/opt/saymon-agent/conf/logback-daemon-debug.xml' -> '/opt/saymon-agent.update/conf/logback-
daemon-debug.xml'
Swap /opt/saymon-agent -> /opt/saymon-agent.backup
Swap /opt/saymon-agent.update -> /opt/saymon-agent
Restart SAYMON Agent service.
SAYMON Agent update completed.
```

4.1.11.3. Linux

Пакетный менеджер

Для обновления агента, установленного через пакетные менеджеры нужно получить пакет с новой версией агента для своего пакетного менеджера (`.deb` для систем Debian/Ubuntu и `.rpm` для Red Hat/Fedora) и установить его на нужном устройстве:

deb

```
$wget https://downloads.cpult.ru/agent/v5.1.1/saymon-agent_5.1.1-cd6973ea_amd64.deb
$sudo dpkg -i saymon-agent_5.1.1-cd6973ea_amd64.deb
```

rpm

```
$wget https://downloads.cpult.ru/agent/v5.1.1/saymon-agent_5.1.1-cd6973ea_amd64.deb
$dnf install saymon-agent-5.1.1-cd6973ea.x86_64.rpm
```

При успешном обновлении пакетов, в консоль выведутся следующие сообщения:

deb

```
(Reading database ... 37680 files and directories currently installed.)
Preparing to unpack saymon-agent_5.1.1-cd6973ea_amd64.deb ...
Stop service
Removed "/etc/systemd/system/multi-user.target.wants/saymon-agent.service".
Backup the configuration files
saymon:x:996:
saymon:x:999:996::/opt/saymon-agent:/sbin/nologin
Unpacking saymon-agent (5.1.1-cd6973ea) over (5.1.1-cd6973ea) ...
Setting up saymon-agent (5.1.1-cd6973ea) ...
You can configure /opt/saymon-agent/conf/agent.properties
Backup directory /opt/saymon-agent.backup found. Also you can restore the
configuration from there.
And restart the service
sudo systemctl restart saymon-agent
Start the service
Created symlink /etc/systemd/system/multi-user.target.wants/saymon-agent.service →
/etc/systemd/system/saymon-agent.service.
```

rpm

```
Last metadata expiration check: 0:00:05 ago on Thu 01 Aug 2024 05:13:15 PM MSK.
Dependencies resolved.
================================================================================
================================================================================
==
 Package                                 Architecture                        Version
Repository                               Size
================================================================================
================================================================================
==
Installing:
 saymon-agent                            x86_64                              5.1.1-
cd6973ea                                @commandline                             90
M

Transaction Summary
================================================================================
================================================================================
==
Install  1 Package

Total size: 90 M
Installed size: 186 M
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
1/1
  Running scriptlet: saymon-agent-5.1.1-cd6973ea.x86_64
1/1
saymon:x:988:
saymon:x:990:988::/opt/saymon-agent:/sbin/nologin
Legacy detected. Will be purged for compatibility.
removed '/opt/saymon-agent/systemd/saymon-agent.service'
removed directory '/opt/saymon-agent/systemd'
...
removed '/opt/saymon-agent/saymon-agent.jar'

  Installing       : saymon-agent-5.1.1-cd6973ea.x86_64
1/1
warning: /opt/saymon-agent/conf/agent.properties created as /opt/saymon-
agent/conf/agent.properties.rpmnew
warning: /opt/saymon-agent/conf/logback.xml created as /opt/saymon-
agent/conf/logback.xml.rpmnew

  Running scriptlet: saymon-agent-5.1.1-cd6973ea.x86_64
1/1
Restore configuration.
You can configure /opt/saymon-agent/conf/agent.properties
Backup directory /opt/saymon-agent.backup found. Also you can restore the
configuration from there.
And restart the service
sudo systemctl restart saymon-agent
Start the service
```

Установочный скрипт

Для обновления агента с помощью установочного скрипта нужно получить скрипт для новой версии агента и запустить его на нужном устройстве:

```
$wget https://downloads.cpult.ru/agent/v5.1.1/saymon-agent-5.1.1-linux-x64-installer.sh
$sh saymon-agent-5.1.1-linux-x64-installer.sh
```

Скриптом обновить можно даже агента, который был установлен через пакетный менеджер.

WARNING

Для каждой архитектуры системы существует свой установочный скрипт. При установке и обновлении скриптом неподходящей архитектуры агент будет перезаписан, но сервис не запустится. Исправить некорректную установку можно обновив агент скриптом для нужной архитектуры.

```
Looks like SAYMON Agent has been already installed in /opt/saymon-agent directory.
Do you want to update application with respect to current configurations? [y/N]
y
Unpacking the distribution archive... OK
SAYMON Agent will be updated.
Copy resources from package into /opt/saymon-agent.update
Copy resources from previous installation into /opt/saymon-agent.update
'/opt/saymon-agent/storage' -> '/opt/saymon-agent.update/storage'
'/opt/saymon-agent/storage/registration.json' -> '/opt/saymon-
agent.update/storage/registration.json'
'/opt/saymon-agent/conf/agent.properties' -> '/opt/saymon-agent.update/conf/agent.properties'
'/opt/saymon-agent/conf/logback-daemon.xml' -> '/opt/saymon-agent.update/conf/logback-
daemon.xml'
'/opt/saymon-agent/conf/logback.xml' -> '/opt/saymon-agent.update/conf/logback.xml'
'/opt/saymon-agent/conf/logback-daemon-debug.xml' -> '/opt/saymon-agent.update/conf/logback-
daemon-debug.xml'
Swap /opt/saymon-agent -> /opt/saymon-agent.backup
Swap /opt/saymon-agent.update -> /opt/saymon-agent
Systemd detected.
SAYMON Agent systemd service installed and enabled.
Restart SAYMON Agent service.
SAYMON Agent update completed.
```

4.1.11.4. Обновление через web-интерфейс

NOTE

Через web-интерфейс можно обновлять только агентов одной major-версии. Например, можно обновить агента версии 5.1.1 до агента версии 5.5.0, а агента версии 4.4.71 до версии 5.0.0 – нет.

Установленные и зарегистрированные в системы агенты можно обновлять через web-интерфейс. Для этого, каждый выпуск агента имеет специальный архив обновления. Обычно, этот архив имеет название следующего вида:

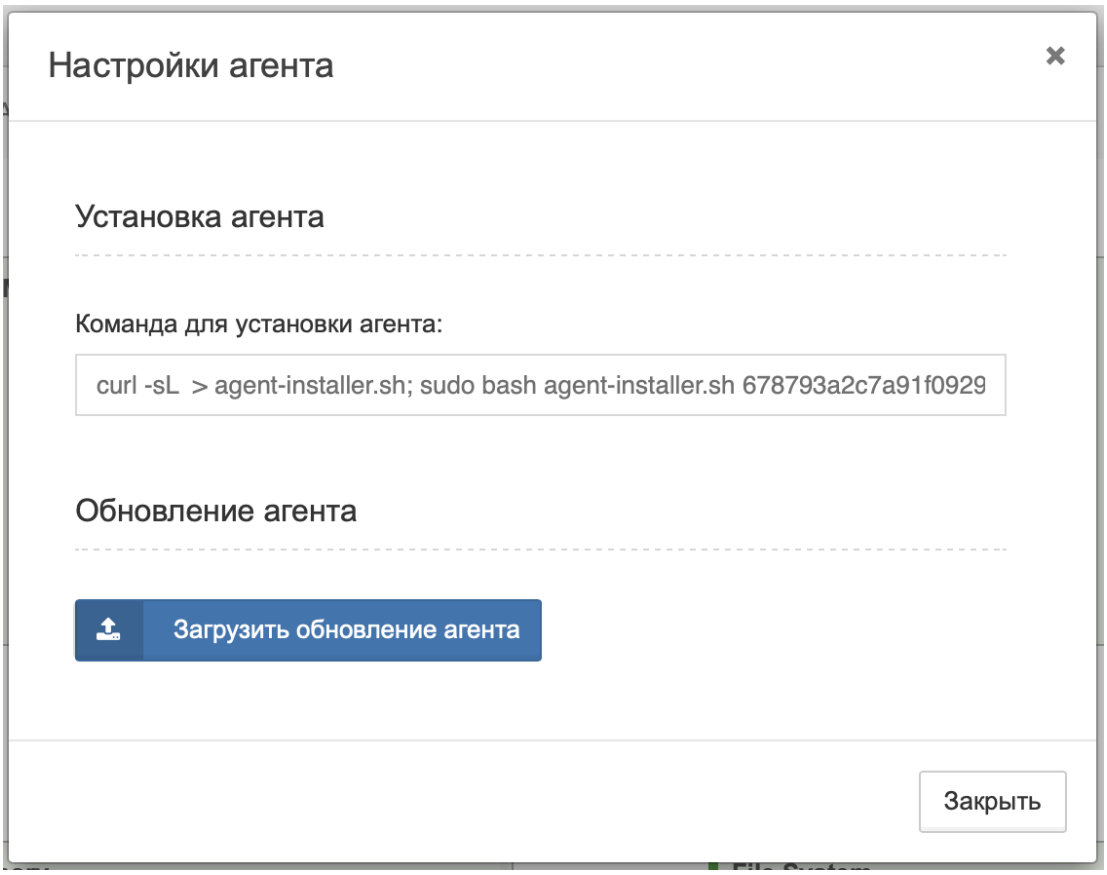
```
saymon-agent-x.x.x-update.zip
```

Для обновления агента через web-интерфейс нужно выполнить следующие шаги:

1. Открыть родительский объект агента в стандартном виде.
2. Навести курсор на объект, представляющий зарегистрированного в системе агента.



3. Нажать на кнопку 🔧, чтобы перейти к окну настройки агента:



4. Нажать на кнопку Загрузить обновление объекта.
5. Выбрать архив с обновлением.

> **TIP**
> Распаковывать архив перед загрузкой не нужно.

После успешной загрузки обновления кнопка Загрузить обновление объекта заменится на кнопку ✔ Обновление успешно загружено.

4.1.12. Метаданные агента

При сборе метрик агент также собирает метаданные. Эти метаданные включают в себя информацию о собранных данных и об агенте, который их собрал.

Они не отображаются в таблице Данные веб интерфейса. Их можно получить запросами Получить метаданные для объекта и Получить метаданные для связи.

4.1.12.1. Поля метаданных

NOTE

В агенте версии 5.0.0 было добавлено новое поле – `headers`, в котором хранится служебная информация.

timestamp

Время когда агент начал выполнять выбранный мониторинговый процесс.

NOTE

Агент версии 5.0.0 и выше также включает эту информация в поле `headers.sendTime`.

entityId

ID объекта, чьи метаданные были получены.

entityType

Тип сущности, чьи метаданные были получены. Для объекта это `1`; для связи – `2`.

taskType

Тип сенсора. Подробную информацию можно найти на странице Встроенные сенсоры.

period

Период между выполнением процессом мониторинга (в миллисекундах).

agentId

ID агента.

NOTE

Агент версии 5.0.0 и выше также включает эту информация в поле `headers.agentId`.

agentVersion

Версия агента.

NOTE

Агент версии 5.0.0 и выше также включает эту информация в поле `headers.version`.

agentBuild

Номер сборки агента.

NOTE

Агент версии 5.0.0 и выше также включает эту информация в поле `headers.version`.

payload

В объекте `payload` хранится мета-информация о каждом поле в собранных агентом данных.

Пример поля `payload` в метаданных сенсора Ping

```
{
...
    "payload": {
        "roundTripMaximum": {
            "changeRate": "ALWAYS"
        },
        "packetsReceived": {
            "changeRate": "SOMETIMES"
        },
        "packetLossPercentile": {
            "type": "PERCENTILE",
            "changeRate": "SOMETIMES"
        },
        ...
    },
...
}
```

payload.changeRate
Частота изменения данных.

Возможные значения:

- `ALWAYS` — дата изменяется после каждого измерения.
- `SOMETIMES` — данные изменяются периодично и будут игнорировать одинаковые значения.
- `NEVER` — данные никогда не обновляются или возможность изменения очень мала.

payload.type
Тип данных, полученных от агента.

Возможные значения:

- `STRING`
- `BOOLEAN`
- `BYTE_QUANTITY`
- `TIMESTAMP`
- `PERCENTILE`
- `ID`
- `BINARY`
- `LIST`

headers

В объекте `headers` хранятся служебные заголовки для трассировки сообщений.

Поле `headers` записывается агентами только начиная с версии 5.0.0.

headers.agentId
ID агента.

Эта информация дублируется в поле `agentId`.

headers.version
Версия агента. Включает в себя как версию агента, так и номер сборки. Эта информацию дублируется в полях `agentVersion` и `agentBuild`.

headers.scheduleTime
Время получения задачи из брокера Kafka.

headers.traceId
Идентификатор выполненной проверки.

headers.startTime
Время начала выполнения проверки.

headers.completeTime
Время завершения выполнения проверки.

headers.sendTime
Время отправки результата в брокер Kafka.

Эта информация дублируется в поле `timestamp`.

5. Мониторинг

5.1. Архитектура мониторинга в системе Центральный Пульт

Центральный Пульт поддерживает два подхода к мониторингу устройств – активный (active) и пассивный (passive).

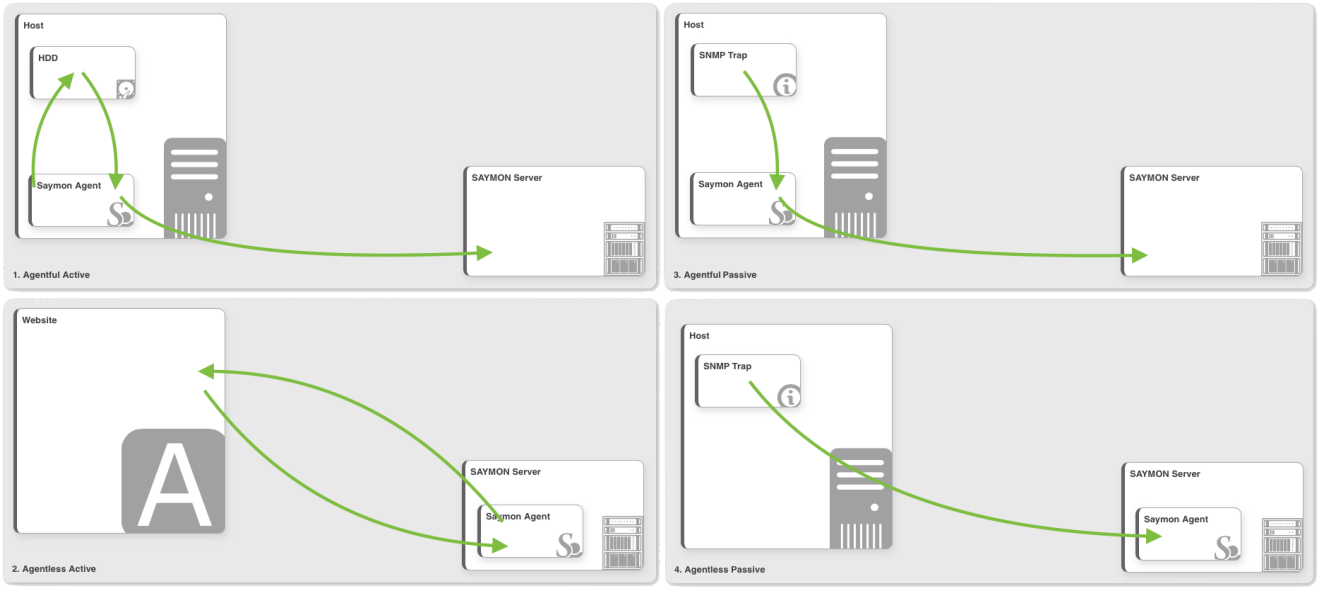
Активный мониторинг подразумевает опрос устройств с определённой периодичностью с целью определения доступности самих устройств и сервисов, которые они предоставляют, а также проверки текущего состояния устройств, например, процент загрузки процессора и дисков.

Пассивный мониторинг подразумевает ожидание от устройств сообщений о событиях, происходящих в системе. Обычно такие сообщения присылаются устройствами с помощью SNMP Trap'ов.

Оба типа мониторинга также можно разделить на две подгруппы по возможности установки агента на наблюдаемый узел или систему: **агентный** (agent) и **безагентный** (agentless).

Агент – это программа, устанавливаемая на виртуальный или физический сервер или иной узел сети и осуществляющая выполнение проверок, сбор и передачу данных серверу. Для организации мониторинга необходимо установить и настроить хотя бы одного агента в системе, например, на сервере системы.

Примеры сочетаний активного/пассивного и агентного/безагентного типов мониторинга:



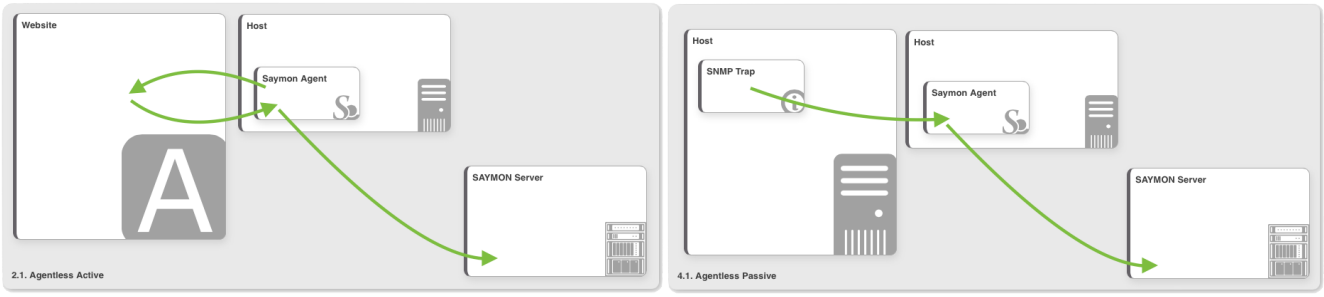
1. **Agentful Active** – активный мониторинг агентом, установленным на хост с проверяемыми параметрами. Агент выполняет локальную проверку состояния жёсткого диска и отправляет результаты на сервер.
2. **Agentless Active** – активный мониторинг агентом, установленным на удаленный хост. Агент, установленный на сервере системы, выполняет проверку работоспособности web-сайта и записывает результаты на сервере.
3. **Agentful Passive** – пассивный мониторинг агентом, установленным на хост с проверяемыми параметрами. Агент получает SNMP Trap от локальной системы и отправляет результаты на сервер.

Agentless Passive – пассивный мониторинг агентом, установленным на удаленный хост. Агент, установленный на сервере системы, получает SNMP Trap от внешней системы и записывает результаты на сервере.

TIP

В некоторых случаях отправка SNMP Trap на локального агента с последующей пересылкой результата на сервер (пример 3) является более оправданной, нежели прямая отправка агенту на удаленный сервер (пример 4), поскольку SNMP Trap использует протокол UDP, не гарантирующий доставку сообщения, в то время как между агентом и сервером устанавливается TCP-соединение.

В случаях безагентного мониторинга использование агента, установленного на сервере системы, не является обязательным – для этого можно использовать любого агента, установленного на доступном в данной части инфраструктуры оборудовании:



5.2. Выбор агента

Для запуска задачи мониторинга нужно выбрать агента, на котором она будет выполняться. В поле **Агент** необходимо выбрать агента, осуществляющему проверку, из выпадающего списка. В списке содержатся все агенты, которые установлены в системе и доступны текущему пользователю.

С помощью фильтра можно найти агента по:

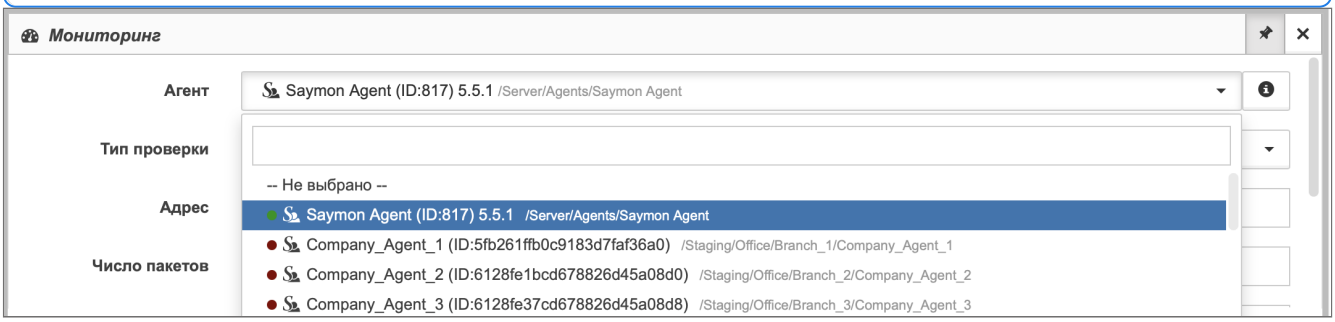
- имени самого агента,
- имени одного из объектов выше в иерархии (от родительского до корневого).

WARNING

Если у агента включен режим мультиродителя, в списке агент будет отображён один раз, в пути будет указан родительский объект с наименьшим ID.

NOTE

Несмотря на то что версия агента отображается в списке, поиск по ней производить нельзя.

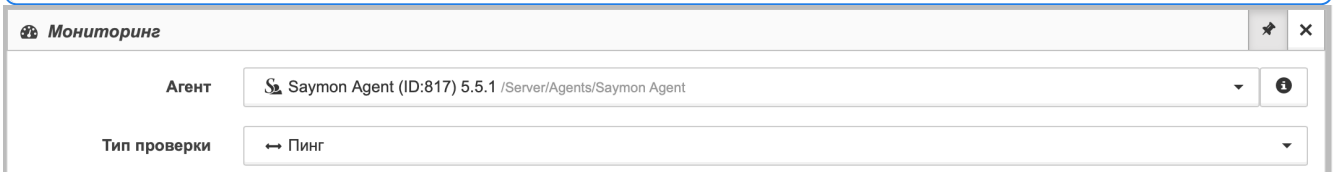


В списке отображаются:

- состояние агента (цвет точки соответствует цвету состояния),
- имя агента,
- ID агента,
- версия агента,
- полный путь до агента в иерархии объектов.

NOTE

Версия агента отображается в этом списке, только если от агента ранее были получены данные.



Кнопка справа от списка открывает окно информации выбранного агента для быстрого просмотра информации о нём.

NOTE

В заголовке окна информации об агенте также указана версия агента.

5.3. Встроенные сенсоры

В этом разделе находится описание встроенных сенсоров, их полей и настроек.

Информацию по настройке пользовательских сенсоров мониторинга можно найти в статье server-configuration:guides/add-custom-probes.html.

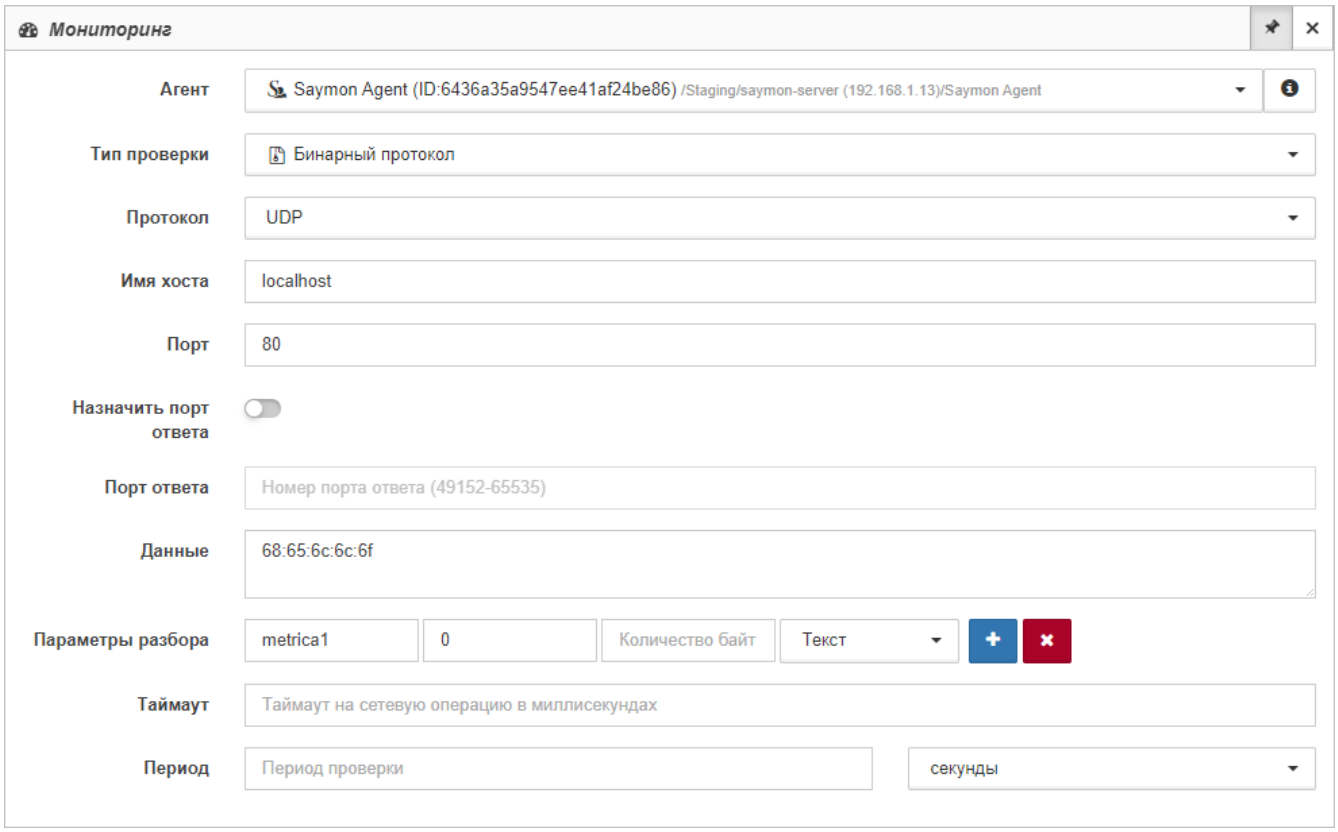
NOTE

Поля и настройки пользовательских сенсоров мониторинга могут отличаться от полей и настроек встроенных сенсоров.

5.3.1. Бинарный протокол

Проверка позволяет отправлять бинарные данные на указанные хост и порт, получать ответ в бинарном формате и трансформировать их в человекочитаемый вид по задаваемым правилам.

5.3.1.1. Пример настройки



Поле	Протокол UDP	Протокол TCP
Имя хоста	Имя или IP-адрес хоста (сервера).	
Порт	Номер порта.	
Назначить порт ответа	Назначать порт получателя.	-
Порт ответа	Номер порта ответа в диапазоне 49152-65535.	-
Данные	Данные для отправки на указанный хост в формате xx:yy:zz.	
Параметры разбора	Параметры отображения результатов выполнения проверки в секции Данные.	
Таймаут	Максимальное время выполнения проверки в миллисекундах.	
Период	Период выполнения проверки (по умолчанию 60 секунд).	

5.3.1.2. Результаты выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:



Поле	Описание
metricaX	Имя метрики, указанное в **Параметрах разбора**

5.3.1.3. Известные ошибки при настройке сенсора

CAUTION

Sensor error! Receive timed out (Время ожидания истекло)

Проверка выполняется дольше, чем указано в поле **Таймаут**.

Необходимо увеличить **таймаут**.

CAUTION

Sensor error! Connection refused! (Отказ в соединении)

Возможные причины:

- некорректно указан номер порта;
- брандмауэр блокирует указанный порт;
- некорректно указано имя хоста;
- указанный хост недоступен из подсети, в которой выполняется проверка.

Проверьте корректность указанного номера порта.

Проверьте правила брэндмауэра и убедитесь, что среди блокирующих правил нет указанного порта.

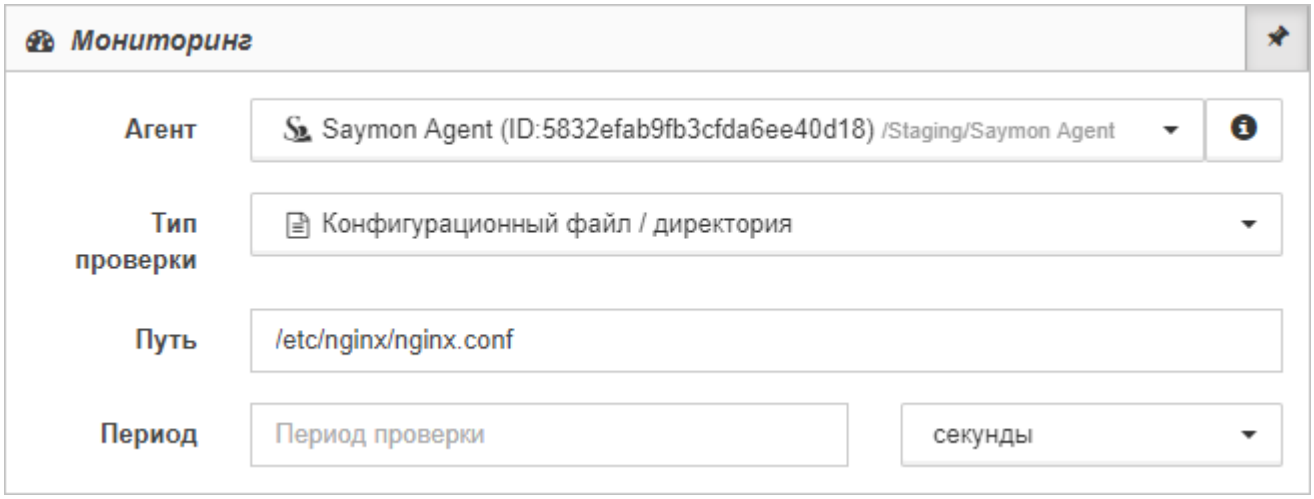
Проверьте корректность указанного имени хоста.

5.3.2. Конфигурационный файл / директория

Проверка позволяет наблюдать за изменениями файлов и папок.

Эту проверку рекомендуется использовать с классом объектов `Configuration File`. В этом случае секция Данные будет заменена секцией Изменения конфигурации, которая предоставляет расширенные возможности для работы с папками и версиями файлов.

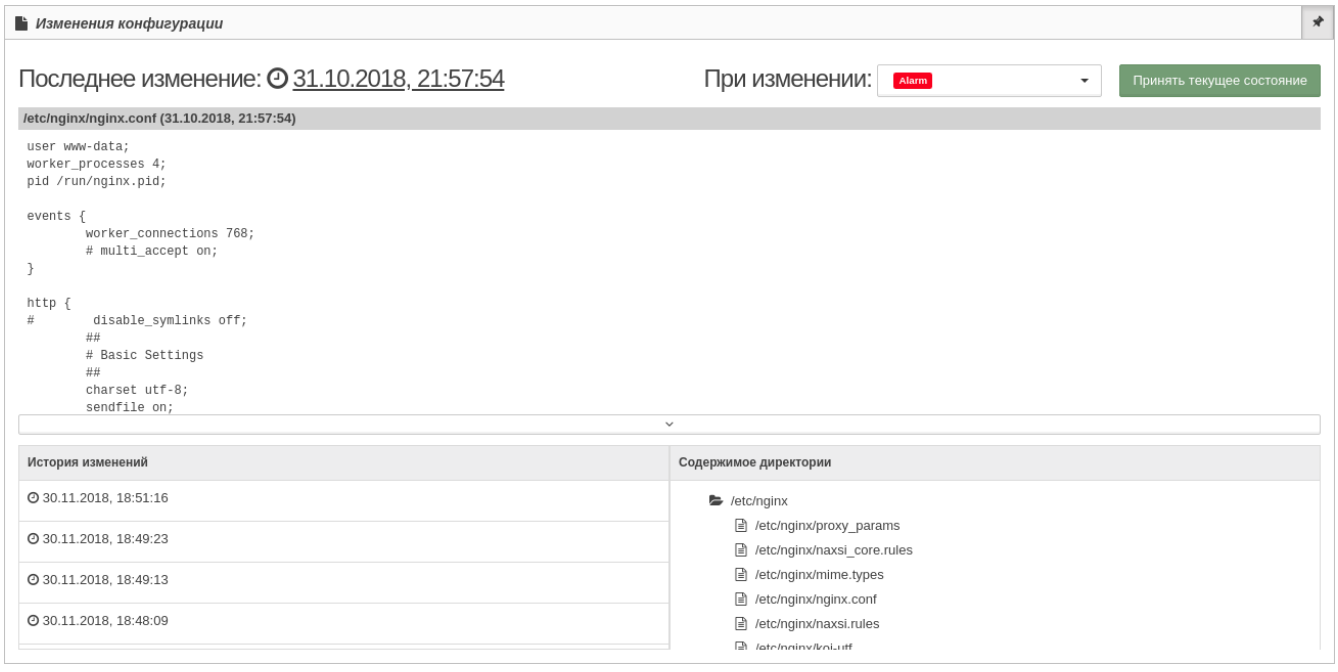
5.3.2.1. Пример настройки



Поле	Описание
Путь	Полный путь к файлу или директории.
Период	Период выполнения проверки (по умолчанию 60 секунд).

5.3.2.2. Результат выполнения

После успешного выполнения проверки в секции Изменения конфигурации появятся дата, время, история изменений и содержимое файла/директории:



5.3.2.3. Известные ошибки при настройке сенсора

CAUTION

Sensor error! /boot/System.map-3.19.0-25-generic (Permission denied)

Нет доступа к указанному файлу/директории.

5.3.3. Запрос в базу данных

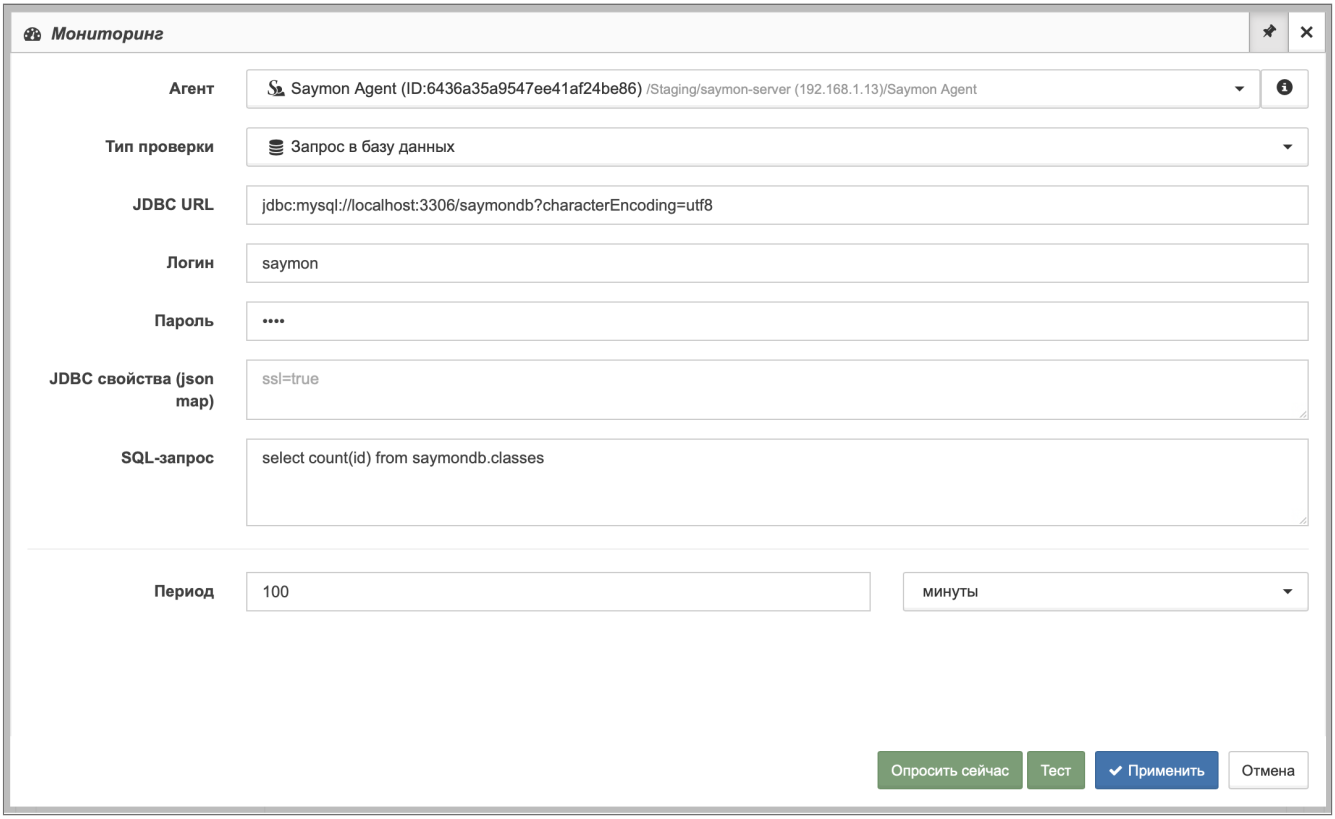
Осуществляет выборку из баз данных:

- MySQL (4.1 - 5.7),
- MS SQL (Microsoft SQL Server 2005 / 2008 / 2008 R2 / 2012 / 2014),
- PostgreSQL (9.x),
- Oracle (9.0 - 11.2),
- HP Vertica.

TIP

Начиная с версии 5.0 агент поддерживает TLS для баз данных PostgreSQL. Инструкцию по его настройке можно найти в статье guides:agent/postgresql-tls.html.

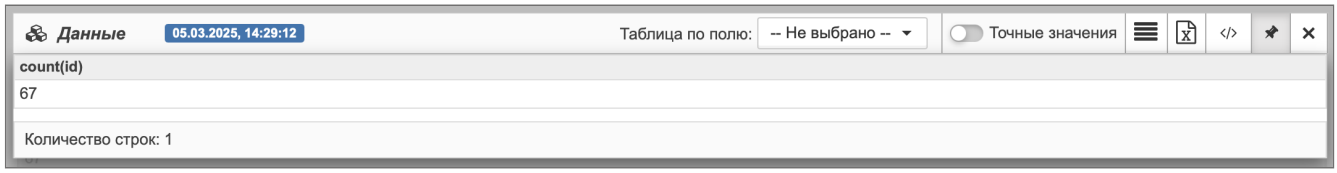
5.3.3.1. Пример настройки



Поле	Описание
JDBC URL	URL базы данных в формате JDBC.
Логин	Имя пользователя БД.
Пароль	Пароль БД.
JDBC свойства (json map)	Свойства JDBC подключения к БД в формате JSON.
SQL-запрос	Текст SQL-запроса. **WARNING** Можно использовать только команду **SELECT**, поле запроса ограничено 1024 символами.
Период	Период выполнения проверки (по умолчанию 60 секунд).

5.3.3.2. Результаты выполнения

После успешного выполнения приведённого выше примера проверки в секции Данные появится следующая информация:



Поле	Описание
count(id)	Количество записей в столбце **ID** из таблицы **States**.

`Количество строк` - дополнительное поле, в котором указана длина массива.

Суммарное количество строк отображается в тех случаях, когда результат проверки или входящие данные представляют собой массив значений.

5.3.3.3. Консольная программа

Выполнение данной проверки в терминале:

```
mysql -u saymon -p
mysql> show databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| saymondb           |
```

```
+--------------------+
2 rows in set (0.00 sec)


mysql> use saymondb
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A


Database changed
mysql> show tables
    -> ;
+--------------------+
| Tables_in_saymondb |
+--------------------+
| class_categories   |
| classes            |
| history            |
| migration          |
| obj_prop_types     |
| object_properties  |
| objects            |
| relation_types     |
| relations          |
| state_history      |
| states             |
+--------------------+
11 rows in set (0.00 sec)


mysql> select count(id) from states;
+-----------+
| count(id) |
+-----------+
|         9 |
+-----------+
1 row in set (0.00 sec)
```

5.3.3.4. Известные ошибки при настройке сенсора

CAUTION

Sensor error! Can not issue data manipulation statements with executeQuery()

Некорректно указан SQL-запрос.

CAUTION

Sensor error! Connection refused (Отказ в соединении)

Необходимо проверить корректность указанного номера порта.

По умолчанию:

- MySQL - 3306;
- PostgreSQL - 5432;
- MS SQL - 1433;
- Oracle - 1521;
- HP Vertica - 5433.

CAUTION

Sensor error! Access denied for user 'say'@'localhost'

Необходимо проверить корректность данных в полях **Логин** и **Пароль**.

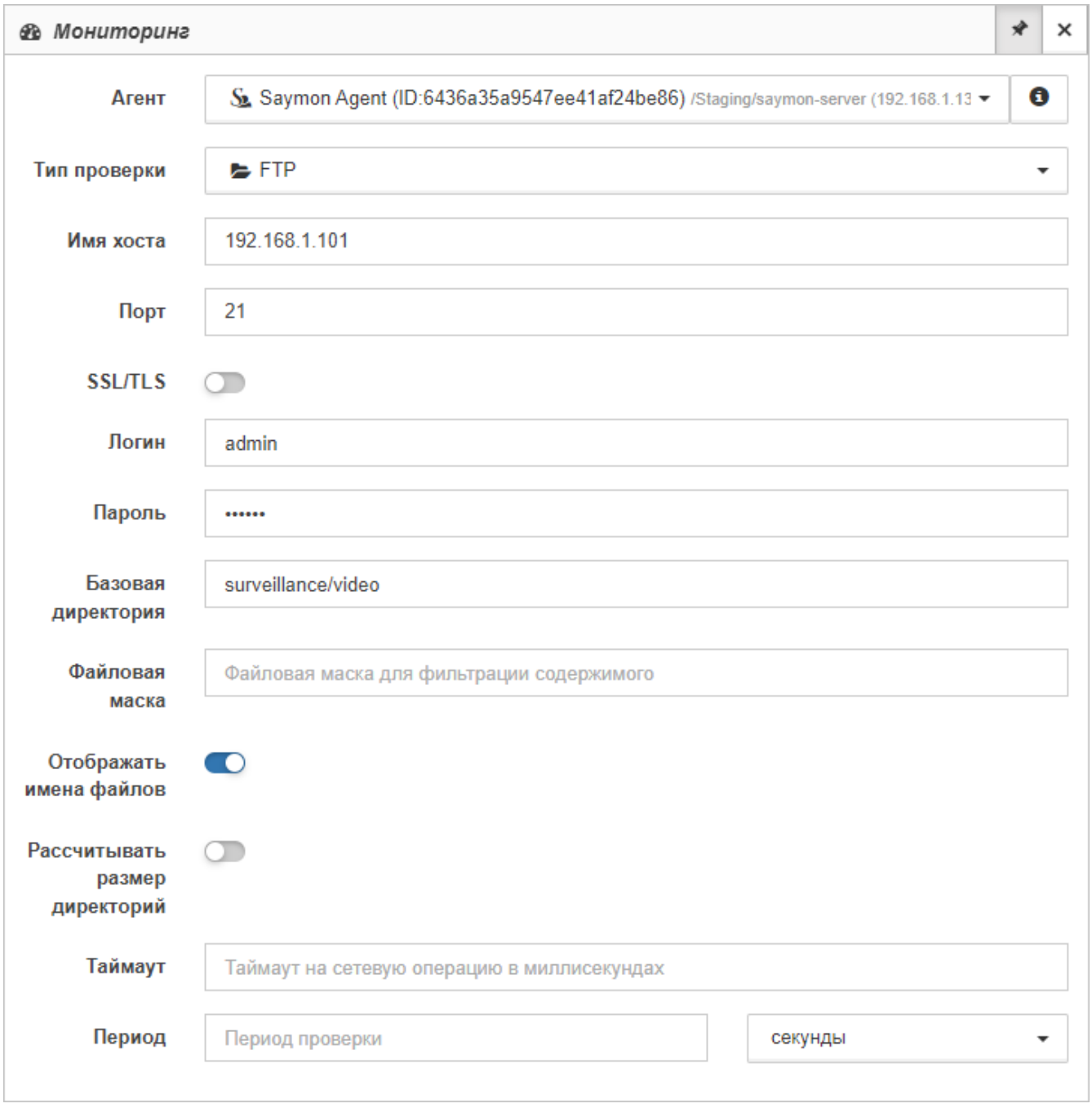
5.3.4. FTP-сенсор

Проверка позволяет подключаться к FTP-директориям и получать данные о размещённых там файлах и папках.

TIP

Эта проверка поддерживает функционал подстановки значений из словарей.

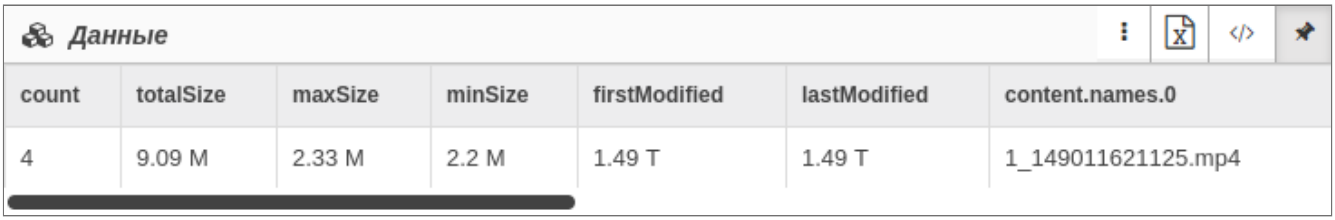
5.3.4.1. Пример настройки



Поле	Описание
Имя хоста	IP-адрес или имя хоста (сервера).
Порт	Номер порта (по умолчанию 21).
SSL/TLS	Защищённое FTP-соединение через SSL/TLS.
Логин	Имя пользователя FTP.
Пароль	Пароль FTP.
Базовая директория	Путь к директории, размещённые в которой папки и файлы доступны для пользователя.
Файловая маска	Файловая маска для фильтрации содержимого директории.
Отображать имена файлов	Отображение в секции Данные имён файлов, хранящихся в директории.
Рассчитывать размер директорий	Отображение в секции Данные суммарного размера вложенных в директорию папок.
Таймаут	Максимальное время выполнения проверки в миллисекундах (по умолчанию 30 секунд).
Период	Период выполнения проверки (по умолчанию 60 секунд).

5.3.4.2. Результаты выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:



Поле	Описание
count	Количество файлов в указанной директории.
totalSize	Размер указанной директории или суммарный размер всех вложенных директорий с файлами, если режим **Рассчитывать размер директорий включен**.
maxSize	Максимальный размер файла в директории.
minSize	Минимальный размер файла в директории.
firstModified	Дата первого изменения директории.
lastModified	Дата последнего изменения директории.
content.names.X	Имена файлов, вложенных в директории, если включен режим **Отображать имена файлов**.

5.3.4.3. Консольная программа

Выполнение проверки в терминале:

```
ftp
ftp> open
(to) 192.168.1.101 21
Connected to 192.168.1.101.
220 (vsFTPd 3.0.2)
Name (192.168.1.101:user1): mbox
331 Please specify the password.
Password:
230 Login successful.
Remore system type is UNIX.
Using Binary mode to transfer files.
ftp> ls 2V422-ZWEQ4-SJMFZ-ZR849-U8461/2017_03_21_17_10_02
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rwxrwxrwx   1 1006    33     2299570 Mar 21 2017 1_149011621125.mp4
-rwxrwxrwx   1 1006    33     2327385 Mar 21 2017 1_14901162199.mp4
-rwxrwxrwx   1 1006    33     2258241 Mar 21 2017 2_149011621123.mp4
-rwxrwxrwx   1 1006    33     2203334 Mar 21 2017 2_149011621988.mp4
226 Directory send OK.
```

5.3.4.4. Известные ошибки при настройке сенсора

CAUTION

Sensor error! Authentication failed.

Ошибка аутентификации.

Проверьте корректность данных, указанных в полях **Логин** и **Пароль**.

CAUTION

Sensor error! No route to host (Host unreachable).

Указанный хост не отвечает.

Возможные причины:

- некорректно указано имя хоста;
- указанный хост недоступен из подсети, в которой выполняется проверка;
- проверяемый хост (сервер) выключен.

Проверьте корректность указанного имени хоста.

CAUTION

Sensor error! Connection refused (Connection refused).

Отказ в соединении.

Возможные причины:

- некорректно указан номер порта;
- брандмауэр блокирует указанный порт.

Проверьте корректность указанного номера порта (по умолчанию 21).

Проверьте правила брандмауэра и убедитесь, что среди блокирующих правил нет указанного порта.

5.3.5. HTTP-запрос

Проверка позволяет выполнять HTTP-запросы:

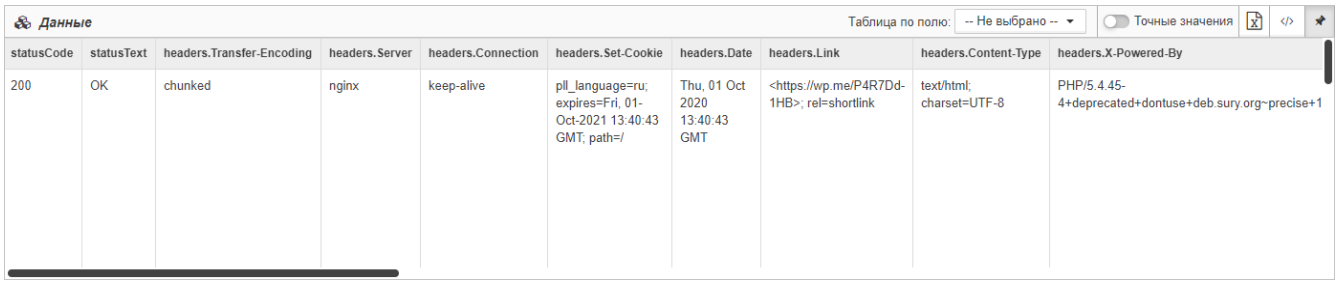
- GET,
- POST,
- HEAD,
- PUT,
- PATCH,
- DELETE.

5.3.5.1. Пример настройки

Поле	Описание
Тип запроса	Тип выполняемого запроса:GET,POST,HEAD,PUT,PATCH,DELETE.
URL	Адрес проверяемого ресурса.
HTTP-заголовки	Заголовки запроса, дополнительно отправляемые с запросом.
Тело запроса	Данные, отправляемые с запросами **POST**, **PUT** и **PATCH**.
Присылать тело ответа	Отключить тело ответа в случая, когда достаточно проверять только HTTP Status Code или скорость ответа (Response Time).
Формат ответа	Если при выполнении проверки возвращаются данные в форматах JSON или XML, то они будут автоматически преобразованы в табличный вид.
Таймаут	Максимальное время выполнения проверки в миллисекундах.
Период	Период выполнения проверки (по умолчанию 60 секунд).
HTTP аутентификация	Выбор схемы аутентификации для запроса. Поддерживаемые способы – `Нет` (сервер не требует аутентификации), `Авто`, `Basic`, `Digest`, `NTLM` и `Negotiate`.
Логин	Имя пользователя или токен аутентификации.
Пароль	Пароль пользователя. Игнорируется для аутентификации по токену.
Использовать прокси-сервер	Использовать прокси-сервер для отправки запроса.
URL прокси-сервера	Адрес прокси-сервера, который будет использовать сенсор для отправки запроса. Прокси-сервер должен поддерживать схемы аутентификации Basic, Digest, NTLM и Negotiate.
Аутентификация прокси-сервера	Выбор схема аутентификации для прокси-сервера. Поддерживаемые способы – `Нет` (сервер не требует аутентификации), `Авто`, `Basic`, `Digest`, `NTLM` и `Negotiate`.
Имя пользователя прокси-сервера	Имя пользователя или токен аутентификации.
Пароль прокси-сервера	Пароль пользователя. Игнорируется для аутентификации по токену.
Разрешить ненадежный SSL	Разрешить подключение по ненадёжным сертификатам SSL.
Поддерживаемые протоколы SSL	Перечисление поддерживаемых протоколов через запятую. Подключение по другим протоколам будет отклонено. По умолчанию разрешены подключения по всем протоколам.

5.3.5.2. Результат выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:





Поле	Описание
statusCode	Код состояния HTTP.
statusText	Текстовая интерпретация кода состояния HTTP.
headers.Xxx	Заголовки ответа.
body	Тело ответа.
body.Yyy	Тело ответа, разбитое на отдельные поля, если в ответе вернулись данные в форматах JSON или XML.
responseTimeMs	Время отклика ресурса в миллисекундах.

5.3.5.3. Консольная программа

Проверка является аналогом программы **Curl**, выполняемой в терминале:

```
$ curl -sD - -X GET https://saas.saymon.info/

HTTP/2 200
server: nginx/1.13.6
date: Wed, 06 Dec 2017 08:49:33 GMT
content-type: text/html; charset=utf-8
content-length: 669
last-modified: Thu, 30 Nov 2017 09:41:32 GMT
etag: "5a1fd24c-29d"
access-control-allow-origin: *
charset: UTF-8
accept-ranges: bytes

<!doctype html>
<html>
<head>
  <title>Loading...</title>
  <meta charset="utf-8">
  <link rel="shortcut icon" href=../images/favicon.ico?1512034892710" type="image/x-icon">
</head>
<body>
<link rel="stylesheet" href="css/saymon.css?1512034892710">
<link rel="stylesheet" href="saymon.local/css/saymon.local.css?1512034892710">
<style type="text/css" id="pointer-position"></style>
<div class="loader">
  <div class="bounce bounce-1"></div>
  <div class="bounce bounce-2"></div>
  <div class="bounce bounce-3"></div>
</div>
<div class="js-main-region main-region"></div>
  <script data-main="js/main" src="js/main-built.js?1512034892710"></script>
</body>
</html>
```

```
$ curl -sD - -X POST https://saas.saymon.info/node/api/users/session -H "Content-Type:
application/json" -d '{"login":"demo", "password":"demo"}'

HTTP/2 200
server: nginx/1.13.6
date: Wed, 06 Dec 2017 08:49:56 GMT
content-type: application/json
content-length: 38
set-cookie: sid=72174edb-5c18-49f2-809f-cd685ea17638; Path=/
access-control-allow-origin: *

"72174edb-5c18-49f2-809f-cd685ea17638"
```

5.3.5.4. Известные ошибки при настройке сенсора

CAUTION

Sensor error! Prime size must be multiple of 64, and can only range from 512 to 1024 (inclusive)

CAUTION

Sensor error! unable to find valid certification path to requested target

CAUTION

Sensor error! Received fatal alert: handshake_failure

Указанные проблемы возникают при использовании агента в связке с устаревшими версиями Java при проверке https-ресурсов.

Есть два способа решения:

1. Обновить Java, установленную в операционной системе хоста, где установлен агент.
2. Скачать и установить последнюю версию агента, где встроена необходимая версия Java.

CAUTION

Sensor error! Target host is not specified

Проверьте адрес, указанный в поле URL – он должен начинаться с протокола:

правильно – **https://**saymon.info

неправильно – saymon.info

5.3.6. JMX-сенсор

WARNING

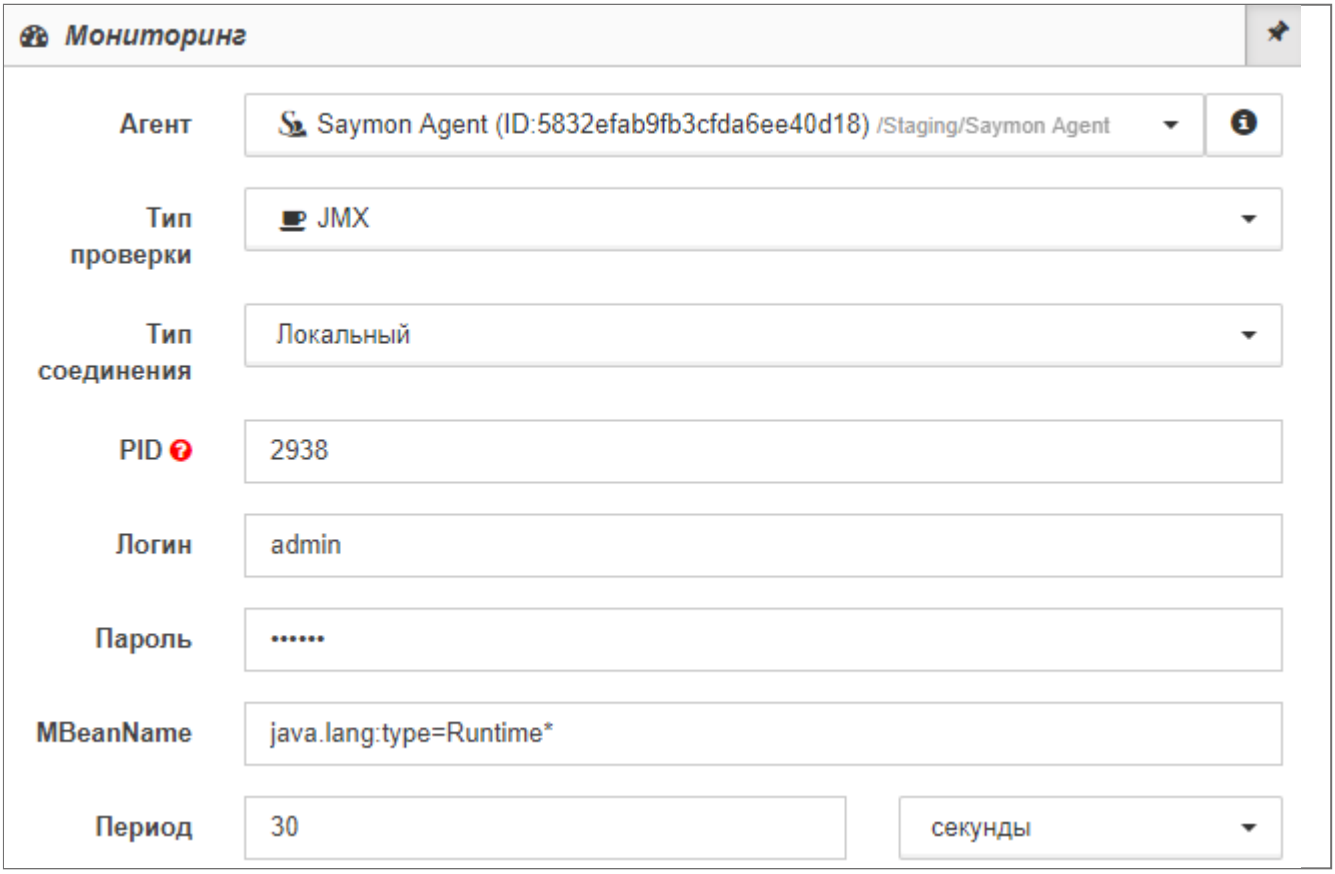
Начиная с версии 5.0.0, агент не поддерживает локальные JMX проверки.

Сенсор позволяет получать данные о работе Java-приложений, поддерживающих JMX – Java Management Extensions.

TIP

Эта проверка поддерживает функционал подстановки значений из словарей.

5.3.6.1. Пример настройки



Поле	При локальном соединении	При удалённом соединении
PID	ID процесса для подключения.	–
Имя хоста	–	IP-адрес или имя хоста.
Порт	–	Номер порта.
Логин	Логин пользователя.	
Пароль	Пароль пользователя.	
MBeanName	Имя Java-объекта, представляющего собой ресурс Java-приложения.	
Период	Период выполнения проверки (по умолчанию 60 секунд).	

5.3.6.2. Результаты выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:

Данные

SpecVendor	ClassPath	InputArguments.0	InputArguments.1
Oracle Corporation	/opt/saymon-agent/saymon-agent.jar	-DSAYMON_AGENT_CONFIG_NAME=/opt/saymon-agent/conf/agent.properties	-DSAYMON_AGENT_TMP_UPDATE_DIRECTORY=/tmp/saymon-agent

NOTE

Набор метрик индивидуален и зависит от проверяемого Java-приложения.

5.3.6.3. Известные ошибки при настройке сенсора

CAUTION

Sensor error! No such process

Процесс отсутствует.

Необходимо проверить корректность ID процесса (PID).

CAUTION

Sensor error! Connection refused

Отказ в соединении.

Возможные причины:

- некорректно указано имя хоста;
- указанный хост недоступен из подсети, в которой выполняется проверка;
- некорректно указан номер порта;
- брандмауэр блокирует указанный порт.

Проверьте корректность указанного имени хоста.

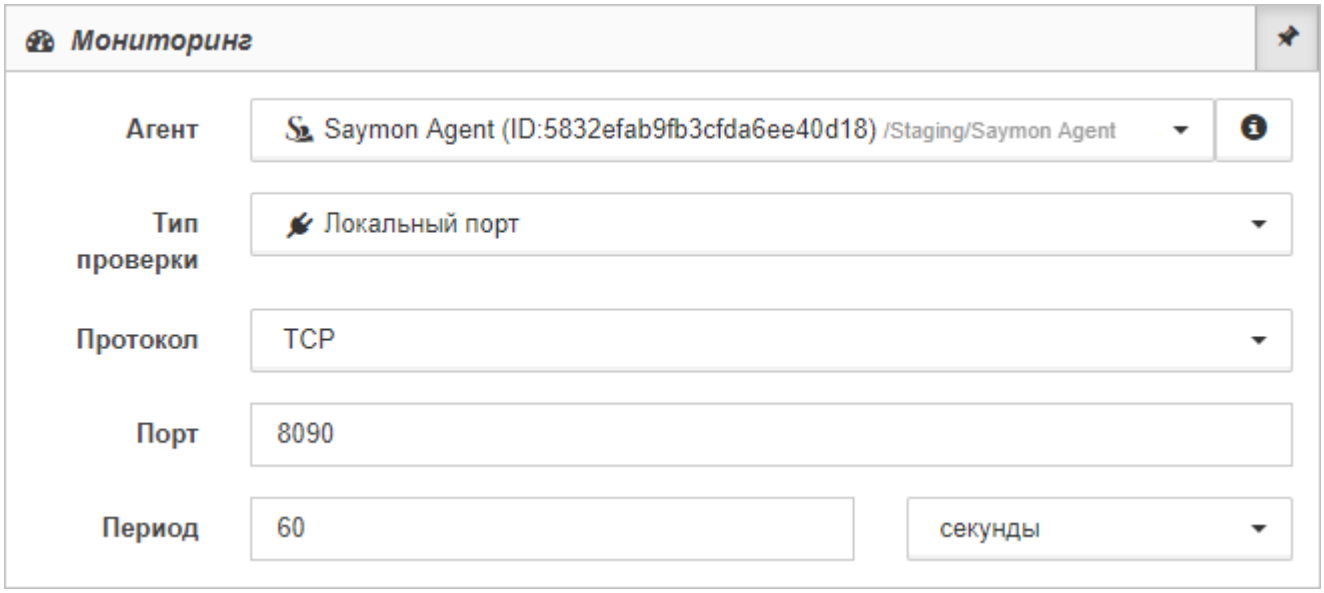
Проверьте корректность указанного номера порта.

Проверьте правила брандмауэра и убедитесь, что среди блокирующих правил нет указанного порта.

5.3.7. Локальный порт

Сенсор проверяет доступность локального порта.

5.3.7.1. Пример настройки



Поле	Описание
Протокол	Протокол проверяемого порта – TCP или UDP.
Порт	Проверяемый порт.
Период	Период выполнения проверки (по умолчанию 60 секунд).

5.3.7.2. Результат выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:



Поле	Описание
success	Результат проверки: • true – порт доступен; • false – порт недоступен.
listenAddress	Адрес, на котором используется проверяемый порт.
processId	ID процесса, который использует проверяемый порт.

5.3.7.3. Консольная программа

Эта проверка является аналогом программы **Netstat**, выполняемой в терминале:

```
$ netstat -anp | grep 8090 | grep LISTEN
tcp        0      0 127.0.0.1:8090          0.0.0.0:*               LISTEN      5563
```

5.3.7.4. Известные ошибки при настройке сенсора

CAUTION

Sensor error! [net.sf.oval.ConstraintViolation: must be between 1 and 65535 inclusive]

Значение указанного порта выходит за пределы возможного диапазона от 1 до 65535.

В данном случае необходимо проверить корректность данных, указанных в поле **Порт**.

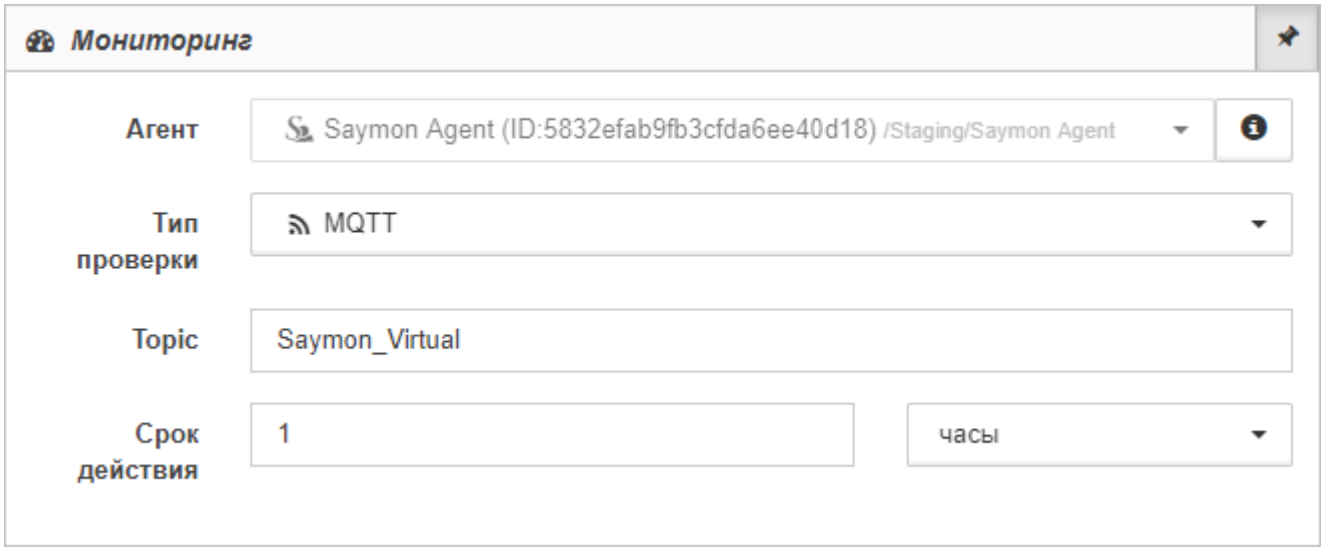
5.3.8. MQTT-сенсор

Проверка позволяет подписаться на топик и получать данные от устройств, поддерживающих работу по протоколу MQTT.

NOTE

Подключение сервера к MQTT-брокеру осуществляется в разделе **MQTT** конфигурационного файла сервера.

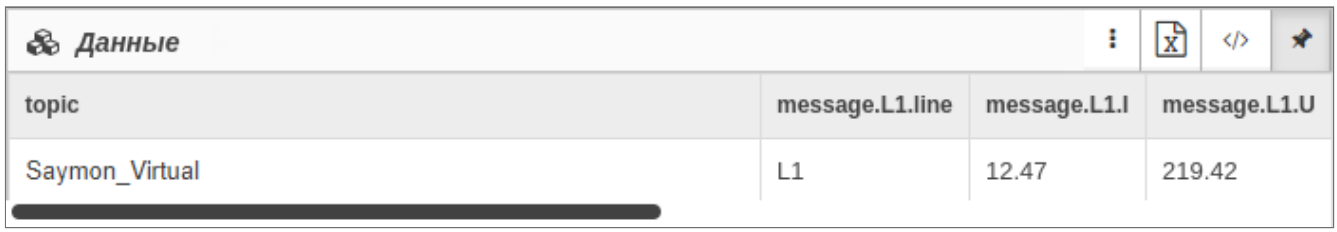
5.3.8.1. Пример настройки



Поле	Описание
Topic	MQTT-топик, на который подписан сервер
Срок действия	Период, в течение которого актуальны полученные данные (0 - без ограничения).

5.3.8.2. Результат выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:



Поле	Описание
topic	MQTT-topic, указанный в настройках сенсора.
message.X.Y	Сообщение, полученное от MQTT-брокера.

5.3.8.3. Консольная программа

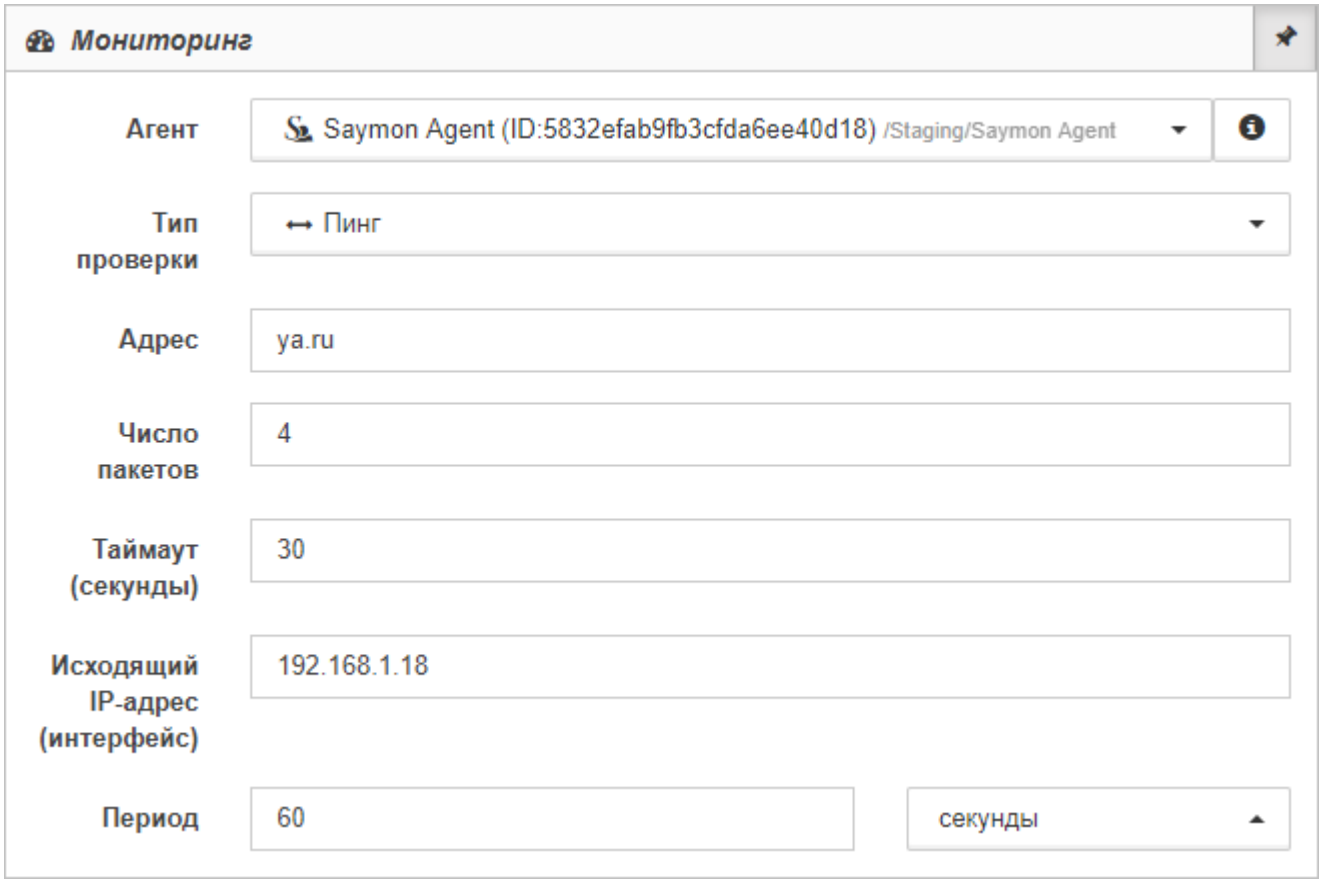
Проверка является аналогом работы программы **mosquitto_sub**, выполняемой в терминале:

```
$ mosquitto_sub -v -t Saymon_Virtual
Saymon_Virtual {"L1":{"line":"L1","I":"12.47","U":"219.42"},"L2":
{"line":"L2","I":"3.59","U":"221.31"},"L3":{"line":"L3","I":"5.18","U":"219.09"}}
```

5.3.9. Пинг-сенсор

Сенсор выполняет команду **ping** до указанного IP-адресу или имени хоста.

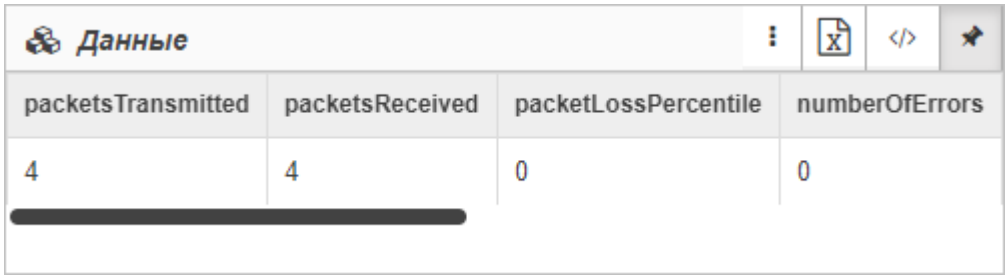
5.3.9.1. Пример настройки

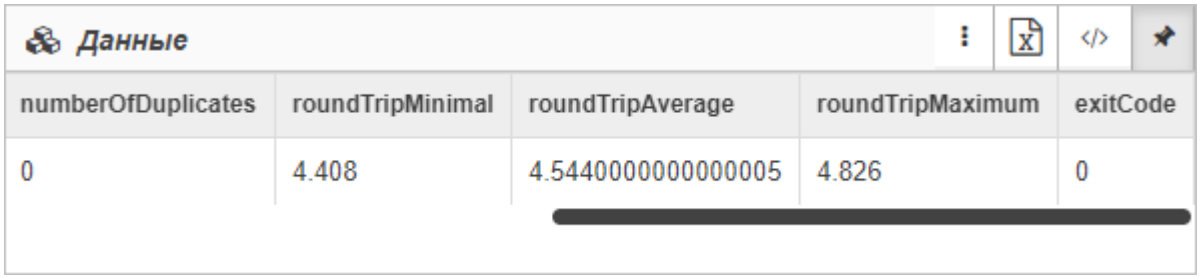


Поле	Описание
Адрес	IP-адрес или имя хоста, который необходимо проверять пингом.
Число пакетов	Число отправляемых пакетов (по умолчанию 4).
Таймаут (секунды)	Максимальное время выполнения проверки (по умолчанию 30).
Исходящий IP-адрес (интерфейс)	Имя сетевого интерфейса или IP-адреса, с которого осуществляется проверка.
Период	Период выполнения проверки (по умолчанию 60 секунд).

5.3.9.2. Результат выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:





Поле	Описание
packetsTransmitted	Количество переданных пакетов.
packetsReceived	Количество полученных пакетов.
packetLossPercentile	Процентиль потерь пакетов.
numberOfErrors	Количество ошибок.
numberOfDuplicates	Количество дубликатов.
roundTripMinimal	Минимальное время приёма-передачи (round-trip time).
roundTripAverage	Среднее время приёма-передачи (round-trip time).
roundTripMaximum	Максимальное время приёма-передачи (round-trip time).
exitCode	Код завершения выполнения проверки (0 – без ошибок).

5.3.9.3. Консольная программа

Эта проверка является аналогом программы **Ping**, выполняемой в терминале:

```
$ ping -I 192.168.1.18 ya.ru

PING ya.ru (87.250.250.242) from 192.168.1.18 : 56(84) bytes of data.

64 bytes from ya.ru (87.250.250.242): icmp_seq=1 ttl=55 time=11.8 ms
64 bytes from ya.ru (87.250.250.242): icmp_seq=2 ttl=55 time=14.7 ms
64 bytes from ya.ru (87.250.250.242): icmp_seq=3 ttl=55 time=11.4 ms
64 bytes from ya.ru (87.250.250.242): icmp_seq=4 ttl=55 time=11.5 ms

--- ya.ru ping statistics ---

8 packets transmitted, 8 packets received, 0.0% packet loss

round-trip min/avg/max/stddev = 11.462/12.066/14.973/1.545 ms
```

5.3.9.4. Известные ошибки при настройке сенсора

CAUTION

Sensor error! Ping exited with error, exit code: 1

Указанный адрес не отвечает (Request timeout).

Возможные причины:

- некорректно указан адрес;
- указанный адрес недоступен из подсети, в которой выполняется проверка;
- проверяемый хост (сервер) выключен;
- проверяемый хост (сервер) или адрес не отвечает, так как на нём отключен ответ на ICMP-запросы.

Проверьте корректность указанного адреса.

Проверьте доступность указанного адреса, выполнив аналогичную команду **ping** в терминале хоста, на котором установлен агент, выполняющий проверку.

CAUTION

Sensor error! Ping exited with error, exit code: 2

Указанный адрес не может быть найден (No route to host).

Возможные причины:

- некорректно указан адрес;
- указанный адрес недоступен из подсети, в которой выполняется проверка;
- некорректно указан исходящий адрес;
- указание префиксов протоколов "http://" или "https://" в поле **Адрес**.

Проверьте корректность указанного проверяемого адреса, исходящего адреса.

Проверьте доступность указанного адреса, выполнив аналогичную команду **ping** в терминале хоста, на котором установлен агент, выполняющий проверку.

Не указывайте префиксы протоколов в поле **Адрес**.

CAUTION

Sensor error! The process was terminated by timeout (Ping exited with error, exit code: 143)

Проверка выполняется дольше, чем указано в поле **Таймаут**.

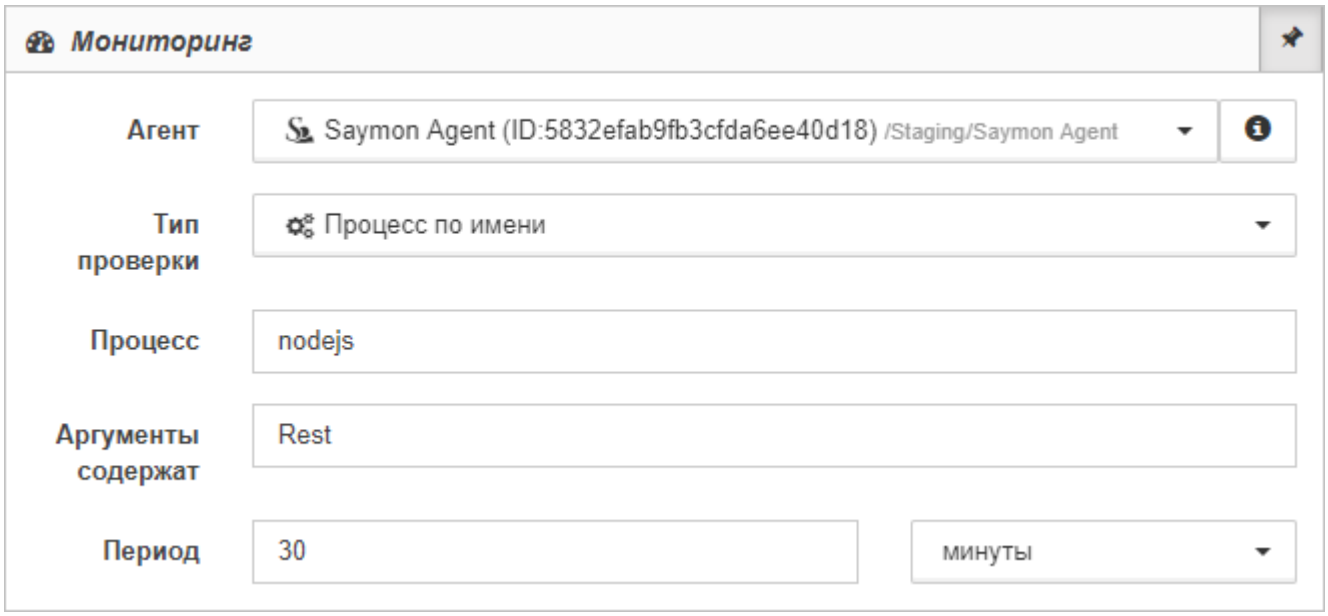
Значение в поле **Таймаут** должно превышать **Число пакетов**, поскольку пакеты отправляются один раз в секунду, и проверка не успевает выполниться.

В этом случае следует уменьшить **Число пакетов** или увеличить **Таймаут**.

5.3.10. Процесс по имени

Процесс по имени возвращает данные по процессу с указанным именем и/или аргументом, запущенному в операционной системе.

5.3.10.1. Пример настройки



Поле	Описание
Процесс*	Имя процесса для поиска. Поиск осуществляется по полному совпадению значения с именем процесса. Например, для **nodejs** - имя процесса совпадает с **nodejs**.
Аргументы содержат*	Поиск осуществляется по частичному совпадению значения хотя бы с одним аргументом процесса. Например, для **Rest** - один из аргументов любого процесса содержит **Rest**.
Период	Период выполнения проверки (по умолчанию 60 секунд).

* Для настройки мониторинга достаточно заполнить одно из полей.

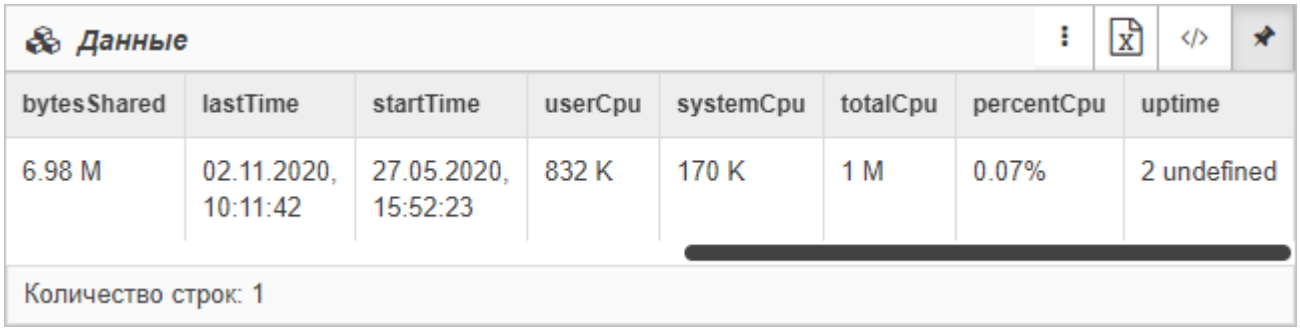
При заполнении обоих полей поиск осуществляется по обоим условиям с применением логического оператора `И`.

Пример: имя процесса совпадает с **nodejs** `И` один из аргументов найденных процессов содержит **Rest**.

5.3.10.2. Результат выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:





Поле	Описание
arguments.xxx	Аргументы, с которыми был запущен данный процесс.
bytesResident	Показывает, сколько физической памяти использует процесс. Соответствует колонке **%MEM** утилит **ps** и **top** - процент использования оперативной памяти данным процессом.
bytesShared	Количество разделяемой памяти, которое используется процессом. Отображает количество памяти, которая потенциально может быть разделена с другими процессами.
bytesVirtual	Используемая виртуальная память или виртуальный "размер" процесса. Показывает общее количество памяти, которое способна адресовать программа в данный момент времени.
lastTime	Время, в которое процесс последний раз выполнялся на CPU.
name	Имя найденного процесса.
nice	Значение приоритета **nice** от -20 до 19.
parentId	ID родительского процесса (PPID).
percentCpu	Количество CPU, используемое данным процессом (или "процент доступного времени процессора, которое использовал запущенный процесс").
processId	ID найденного процесса (PID).
startTime	Время, в которое был запущен процесс.

Поле	Описание
state	Код состояния процесса. • **D** uninterruptible sleep (usually IO) - процесс ожидает ввода-вывода (или другого недолгого события), непрерываемый; • **I** is multi-threaded (using CLONE_THREAD, like NPTL pthreads do) - многопоточный процесс; • **L** has pages locked into memory (for real-time and custom IO) - процесс использует страничную память; • **N** low-priority (nice to other users) - процесс с низким приоритетом, получает ресурсы позже прочих; • **R** running or runnable (on run queue) - процесс выполняется в данный момент или готов к выполнению (состояние готовности); • **s** is the session leader - процесс является лидером сессии; • **S** interruptible sleep (waiting for an event to complete) - процесс в состоянии ожидания (т.е. "спит" менее 20 секунд); • **T** stopped, either by a job control signal or because it is being traced - процесс остановлен или трассируется отладчиком; • **W** paging (not valid since the 2.6.xx kernel) - процесс в стадии "paging", не актуально для ОС с kernel старше 2.6; • **X** dead (should never be seen) - процесс в состоянии завершения; • **Z** defunct ("zombie") process, terminated but not reaped by its parent - завершившийся процесс, код возврата которого пока не считан родителем; • **<** high-priority (not nice to other users) - процесс с высоким приоритетом, получает ресурсы раньше прочих; • **+** is in the foreground process group - процесс запущен в foreground-режиме.
systemCpu	Время CPU, занятое системой.
totalCpu	Общее процессорное время, занятое процессом (сумма userCpu и systemCpu).
tty	Терминал, с которым связан данный процесс.
uptime	Время, в течение которого процесс находится в работе.
userCpu	Время CPU, которое занял пользователь.

`Количество строк` - дополнительное поле, в котором указана длина массива.

Суммарное количество строк отображается в тех случаях, когда результат проверки или входящие данные представляют собой массив значений.

5.3.10.3. Консольная программа

```
$ ps aux | head -1 ; sudo ps aux | grep 'nodejs.*Rest' | grep -v grep

USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
saymon   29050  3.7  4.7 1359092 194040 ?      Sl   Nov30 363:39 /usr/bin/nodejs
/usr/local/saymon/server/actors/lib/forked-actor-worker.js RestServerActor
```

5.3.11. Выполнение программы/скрипта

Выполнение программы/скрипта осуществляет вызов исполняемого файла и возвращает его вывод из потоков **stdout** и **stderr**.

5.3.11.1. Настройка

В этом разделе перечислены настройки для всех подтипов программ/скриптов.

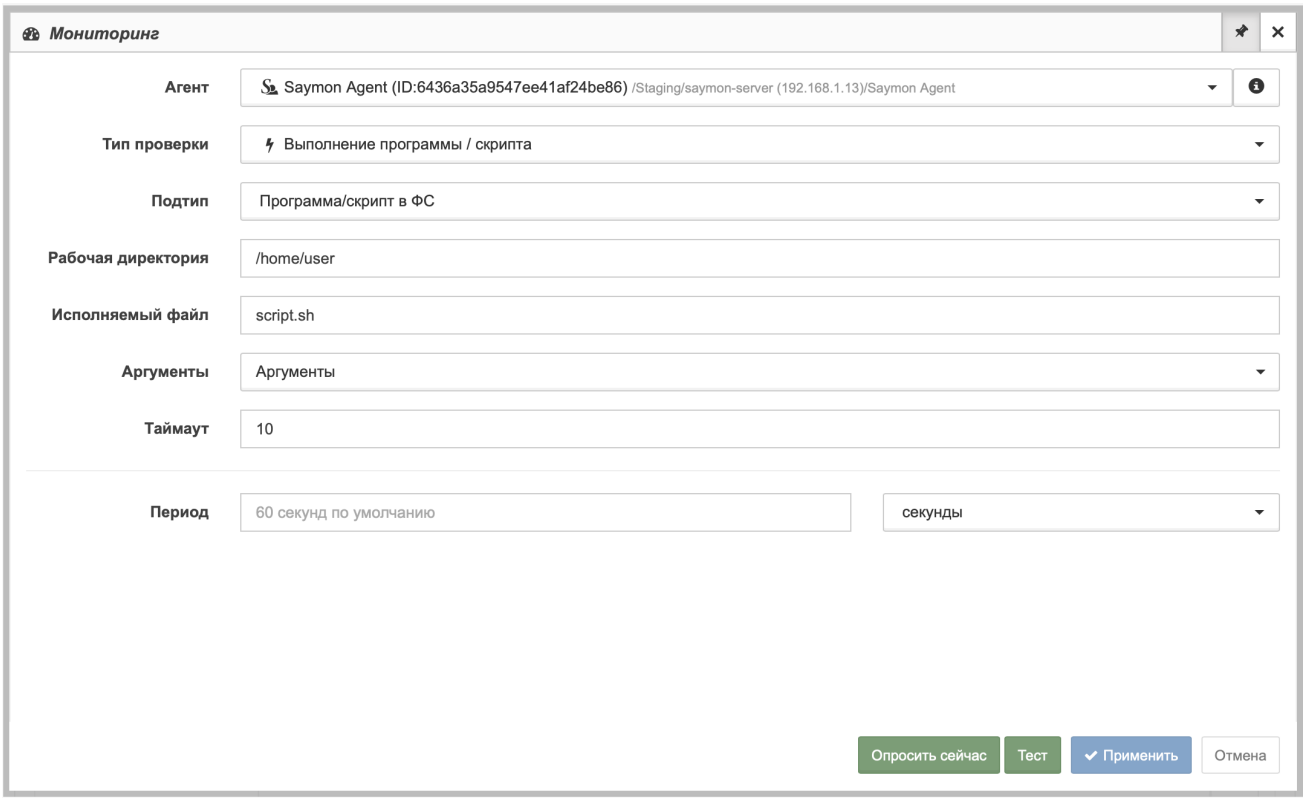


Рис. 7. Пример настроек сенсора с подтипом "Программа\скрипт в ФС"

Подтип

Тип программы/скрипта.

Доступные типы:

- Программа/скрипт в ФС,
- Скрипт из репозитория,
- Скрипт с указанным текстом.

WARNING

Подтип **Скрипт с указанным текстом** доступен только пользователям, у которых есть права на исполнение скриптов. Если же у пользователя нет таких прав, но скрипт был создан другим пользователем, то подтип будет доступен, а скрипт будет в режиме для чтения.

Рабочая директория

WARNING

Этот функционал требует SAYMON Agent v5.5.8 или выше. Агенты меньшей версии всегда будут использовать место установки агента (по умолчанию `/opt/saymon-agent`).

Рабочая директория выполнения скрипта.

Если рабочая директория не указана, то система использует место установки агента (по умолчанию `/opt/saymon-agent`).

Скрипт/Исполняемый файл

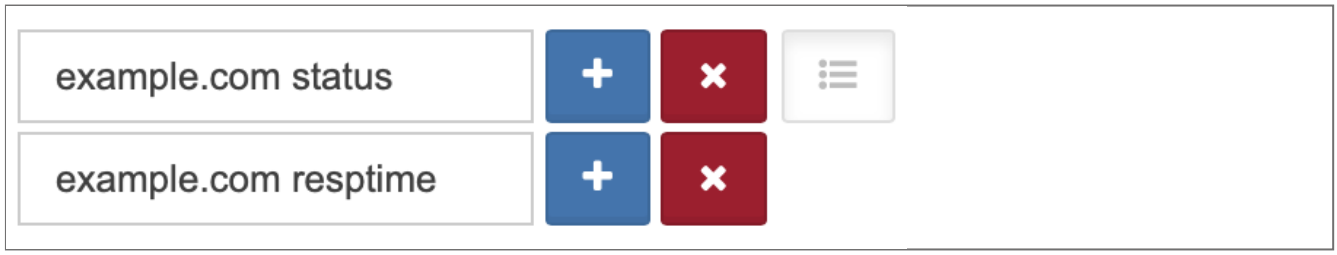
Выбор скрипта, который будет выполняться сенсором. Вид поля зависит от выбранного подтипа.

- Для подтипа **Программа/скрипт в ФС** в поле **Исполняемый файл** указывается имя программы или путь к скрипту,
- Для подтипа **Скрипт из репозитория** в поле **Скрипт** из выпадающего списка выбирается скрипт, заранее сохранённый в репозитории,
- Для подтипа **Скрипт с указанным текстом** в поле **Скрипт** вводится текст скрипта с помощью редактора скриптов.

Аргументы

Аргументы, которые передаются в скрипт.

Если хотя бы один из передаваемых аргументов представляет собой строку с пробелами, то каждый аргумент необходимо указать в отдельном поле:



TIP

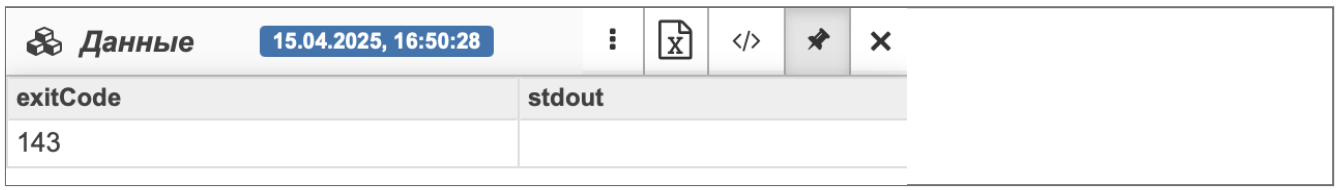
В аргументах можно использовать переменные для передачи значений свойств в скрипт.

Таймаут

Время, в течение которого агент будет ожидать результатов выполнения скрипта или программы с момента её запуска.

Значение таймаута по умолчанию – 30 секунд.

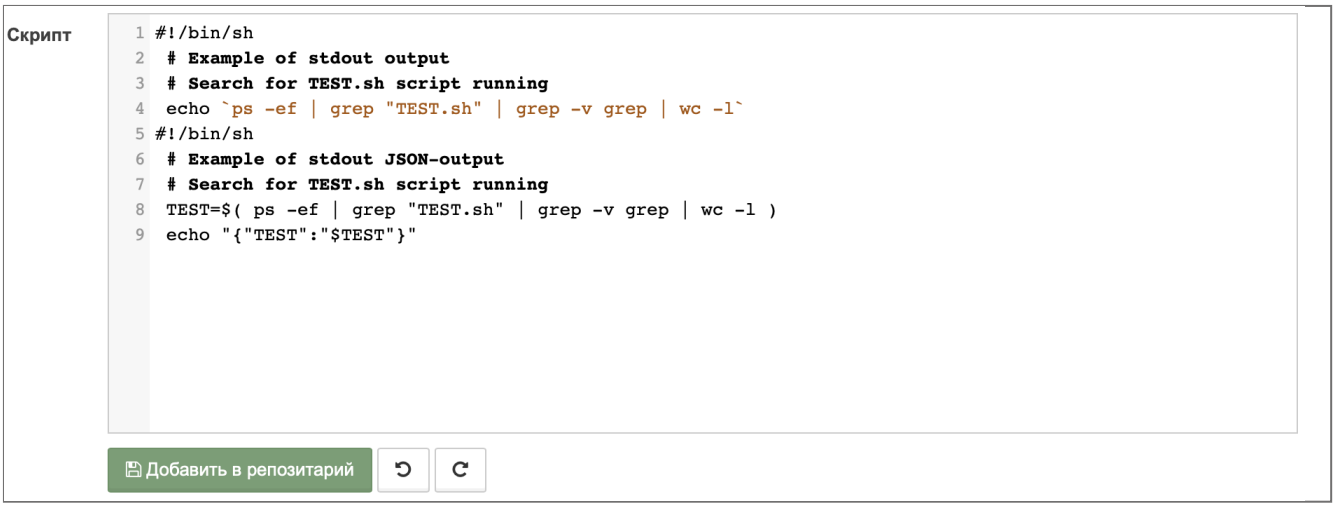
Если выполнение программы или скрипта занимает дольше, чем указанное время таймаута, то объект переходит в состояние WORKING, а проверка возвращает следующие данные:



5.3.11.2. Редактор скриптов

С помощью встроенного редактора скриптов пользовать может задавать скрипты для сенсора **Выполнение программы/скрипта** с подтипом **Скрипт с указанным текстом**. Редактор поддерживает

подсветку кода.



WARNING

Кнопки управления редактором доступны только пользователям, у которых есть права на исполнение скриптов. Если у пользователя нет таких прав, то скрипт будет доступен только в режиме для чтения.

С помощью кнопки Добавить в репозиторий можно добавить редактируемый скрипт в репозиторий. Кнопки и отменяют предыдущее действия и повторяют отменённое действие соответственно.

5.3.11.3. Примеры

Помимо обычных текстовых или числовых данных, из скрипта можно вернуть данные в формате JSON, например:

```
{"cpu": "10", "mem": "20"}
```

В этом случае данные будут распознаны и распределены в таблицу данных по столбцам **cpu** и **mem** со значениями **10** и **20** соответственно:

cpu	mem
10	20

Для вывода многострочной таблицы скрипт должен возвращать данные в ином формате:

```
{
  "host1": {
    "cpu": "10",
    "mem": "20"
  },
  "host2": {
    "cpu": "30",
    "mem": "40"
  }
}
```

В этом случае данные будут распознаны и распределены в таблицу данных по столбцам "host1.cpu", "host1.mem", "host2.cpu" и "host2.mem":

host1.cpu	host1.mem	host2.cpu	host2.mem
10	20	30	40

Если выбрать **stdout** в выпадающем списке **Таблица по полю**, то в секции Данные в заголовке получится многострочная таблица:

cpu	mem
10	20
30	40

Для наглядности можно добавить дополнительный признак **host** в возвращаемые данные:

```
{
  "host1": {
    "host": "1",
    "cpu": "10",
    "mem": "20"
  },
  "host2": {
    "host": "2",
    "cpu": "30",
    "mem": "40"
  }
}
```

Получится таблица:

host	cpu	mem
1	10	20
2	30	40

Ещё один хороший пример:

```
 {
   "MEM": {
     "memoryType": "MEM",
     "bytesTotal": 4130643968,
     "bytesUsed": 3002249216,
     "bytesAvailable": 1128394752,
     "percentUsed": 72.68235266119164
   },
   "SWAP": {
     "memoryType": "SWAP",
     "bytesTotal": 536866816,
     "bytesUsed": 469790720,
     "bytesAvailable": 67076096,
     "percentUsed": 87.50600819403225
   },
   "TOTAL": {
     "memoryType": "TOTAL",
     "bytesTotal": 4667510784,
     "bytesUsed": 3472039936,
     "bytesAvailable": 1195470848,
     "percentUsed": 74.38740040841435
   }
 }
```

Примеры скриптов для *nix:

```
# !/bin/sh
 # Пример с выводом в stdout
 # Поиск запущенного скрипта TEST.sh
 echo `ps -ef | grep "TEST.sh" | grep -v grep | wc -l`

#!/bin/sh
 # Пример с выводом в stdout в формате JSON
 # Поиск запущенного скрипта TEST.sh
 TEST=$( ps -ef | grep "TEST.sh" | grep -v grep | wc -l )
 echo "{"TEST":"$TEST"}"
```

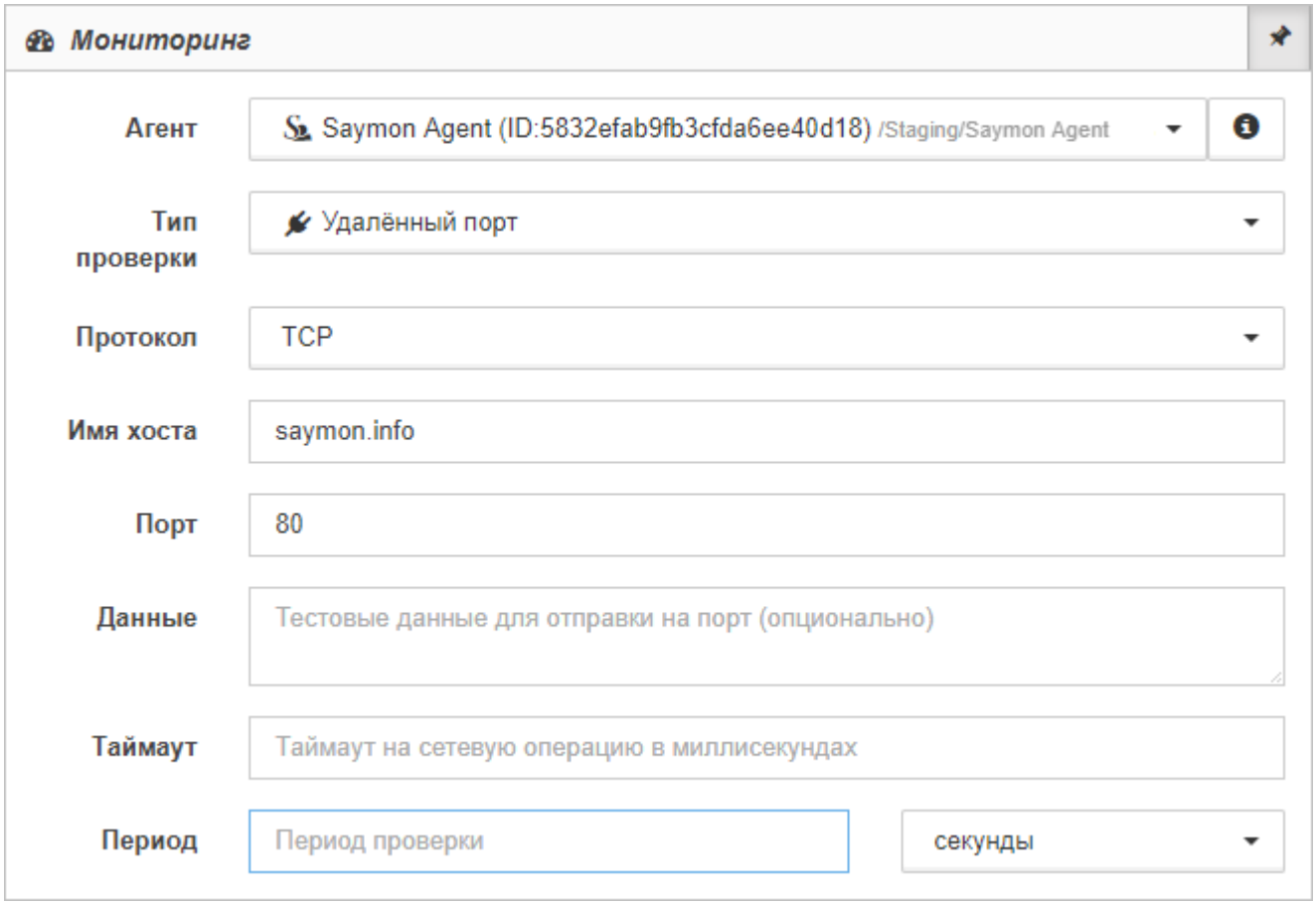
Примеры скриптов для Windows:

```
@echo off
 REM Пример с выводом в stdout
 REM Поиск службы RDP
 for /F "tokens=*" %%i in ('tasklist.exe  /svc ^| find /c "TermService"') do set TERMSRV=%%i
 echo %TERMSRV%
@echo off
 REM Пример с выводом в stdout в JSON формате
 REM Поиск службы RDP и запущенного командного файла test.cmd
 for /F "tokens=*" %%i in ('tasklist.exe  /svc ^| find /c "TermService"') do set TERMSRV=%%i
 for /F "tokens=*" %%i in ('tasklist.exe  /v ^| find /c "test.cmd"') do set TEST=%%i
 echo {"TERMSRV":"%TERMSRV%","TEST":"%TEST%"}
```

5.3.12. Удалённый порт

Сенсор проверяет доступность удалённого порта.

5.3.12.1. Пример настройки



Поле	Описание
Протокол	Протокол проверяемого порта, TCP или UDP.
Имя хоста	IP-адрес или DNS-имя хоста, порт которого необходимо проверять.
Порт	Проверяемый порт.
Данные	Дополнительные тестовые данные, которые можно отправить на порт во время проверки.
Таймаут	Максимальное время выполнения проверки в миллисекундах.
Период	Период выполнения проверки (по умолчанию 60 секунд).

5.3.12.2. Результат выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:



Поле	Описание
success	Результат проверки: • true - порт доступен; • false - порт недоступен.
errorMessage	Сообщения об ошибках выполнения проверки или о причинах недоступности проверяемого порта.

5.3.12.3. Консольная программа

Эта проверка является аналогом программы **Netcat**, выполняемой в терминале:

```
$ netcat -zv saymon.info 80
> Connection to saymon.info 80 port [tcp/http] succeeded!

$ netcat -zvu saymon.info 80
> Connection to saymon.info 80 port [udp/http] succeeded!
```

5.3.12.4. Известные ошибки при настройке сенсора

CAUTION

Sensor error! [net.sf.oval.ConstraintViolation: must be between 1 and 65535 inclusive]

Значение указанного порта выходит за пределы возможного диапазона от 1 до 65535 (Port number is too large).

В данном случае необходимо проверить корректность данных, указанных в поле **Порт**.

CAUTION

messageError - UnknownHostException: http://saymon.info

Указанный хост не может быть найден (getaddrinfo: Name or service not known).

Возможные причины:

- некорректно указано имя хоста;
- указанное имя недоступно из подсети, в которой выполняется проверка;
- указание префиксов протоколов "http://" или "https://" в имени хоста.

Проверьте корректность указанного адреса.

Проверьте доступность указанного хоста, выполнив команду **netcat** в терминале хоста, на котором установлен агент, выполняющий проверку.

Не указывайте префиксы протоколов в поле **Имя хоста**.

5.3.13. SNMP Get

Сенсор выполняет запрос **SNMP Get** по указанному OID (идентификатор объекта в SNMP MIB) заданного хоста.

NOTE

Для функционала **SNMP GETNEXT** и **SNMP WALK** требуется SAYMON Agent 5.5.1.

Также, этот сенсор может выполнять запросы **SNMP GETNEXT** и **SNMP WALK**. Выполняемая операция зависит от формата поля OID:

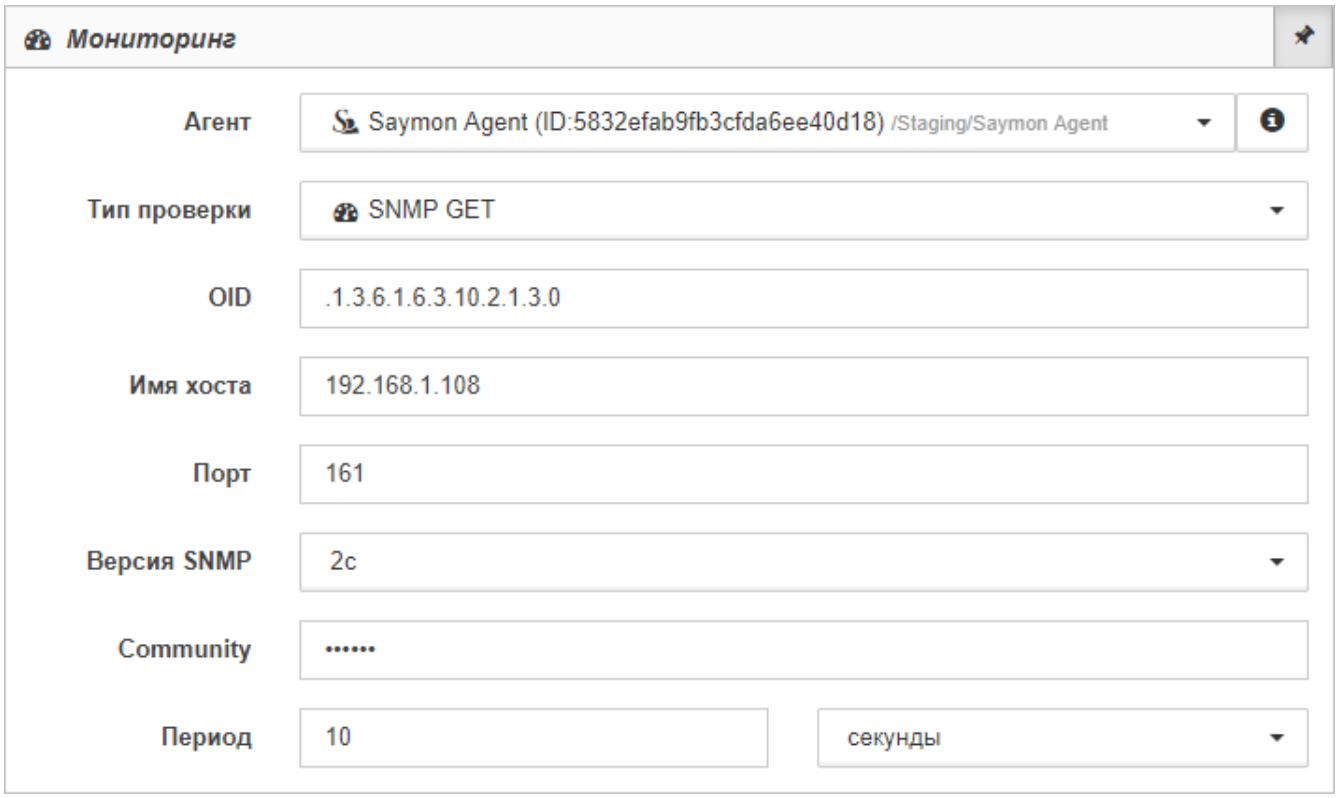
- Если поле OID заканчивается не на точку – выполняется операция **SNMP GET**;
- Если поле OID заканчивается на одну точку – выполняется **SNMP WALK**;
- Если поле OID заканчивается на две точки – выполняется **SNMP GETNEXT**.

TIP

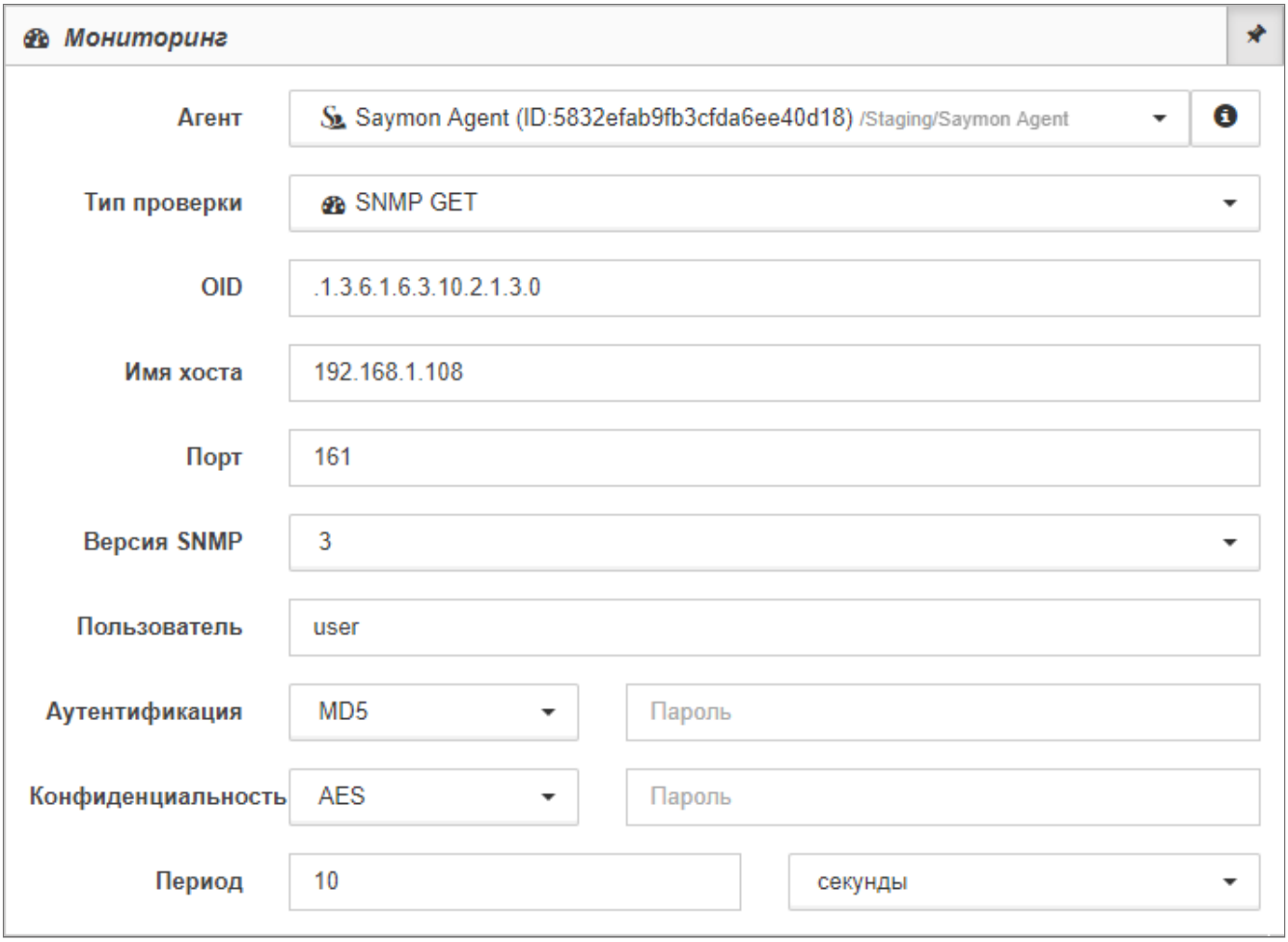
Если задано несколько OID типа **SNMP GETNEXT**, то сенсор обрабатывает только первый. Если задано несколько OID типа **SNMP GET** и **SNMP WALK**, то сенсор выполнит запросы ко всем указанным OID.

5.3.13.1. Пример настройки

Версия SNMP 1/2c:



Версия SNMP 3:


Мониторинг
Агент
Saymon Agent (ID:5832efab9fb3cfda6ee40d18) /Staging/Saymon Agent
Тип проверки
SNMP GET
OID
.1.3.6.1.6.3.10.2.1.3.0
Имя хоста
192.168.1.108
Порт
161
Версия SNMP
3
Пользователь
user
Аутентификация
MD5
Пароль
Конфиденциальность
AES
Пароль
Период
10
секунды

Поле	Описание
OID	Идентификатор объекта, значение которого необходимо получить. Можно задавать несколько OID типов **SNMP GET** и **SNMP WALK** через пробел. При этом, сенсор выполнит запросы ко всем указанным OID.
Имя хоста	IP-адрес или имя хоста, на котором расположен опрашиваемый SNMP-агент.
Порт	Порт, на котором расположен опрашиваемый SNMP-агент.
Версия SNMP	Версия используемого SNMP-протокола: • 1 (RFCs 1155-1157), • 2c (RFCs 1901-1908), • 3 (RFCs 2571-2574).
Community (SNMP v1/v2c)	Значение community (пароля), используемое для аутентификации транзакций.
Пользователь(SNMP v3)	Имя пользователя, используемое для аутентификации.
Аутентификация(SNMP v3)	Способ аутентификации. Возможные значения: • Нет (не использовать аутентификацию); • MD5 (нужно также указать пароль); • SHA (нужно также указать пароль).
Конфиденциальность(SNMP v3)	Способ шифрования. Возможные значения: • Нет (не использовать аутентификацию); • DES (нужно также указать пароль); • AES (нужно также указать пароль).
Период	Период выполнения проверки (по умолчанию 60 секунд).

5.3.13.2. Результат выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:



Поле	Описание
<Номер запрошенного OID>	Значение запрошенного SNMP-объекта.

5.3.13.3. Консольная программа

Эта проверка является аналогом программы **SnmpGet**, выполняемой в терминале:

```
$ snmpget -On -v 2c -c public 192.168.1.108:161 .1.3.6.1.6.3.10.2.1.3.0

> .1.3.6.1.6.3.10.2.1.3.0 = INTEGER: 61362 seconds
```

5.3.14. SNMP Trap

Агент может выступать в роли получателя SNMP Trap'ов и отправлять данные на сервер. Принимаемые сервером данные от всех агентов можно увидеть в Журнале событий.

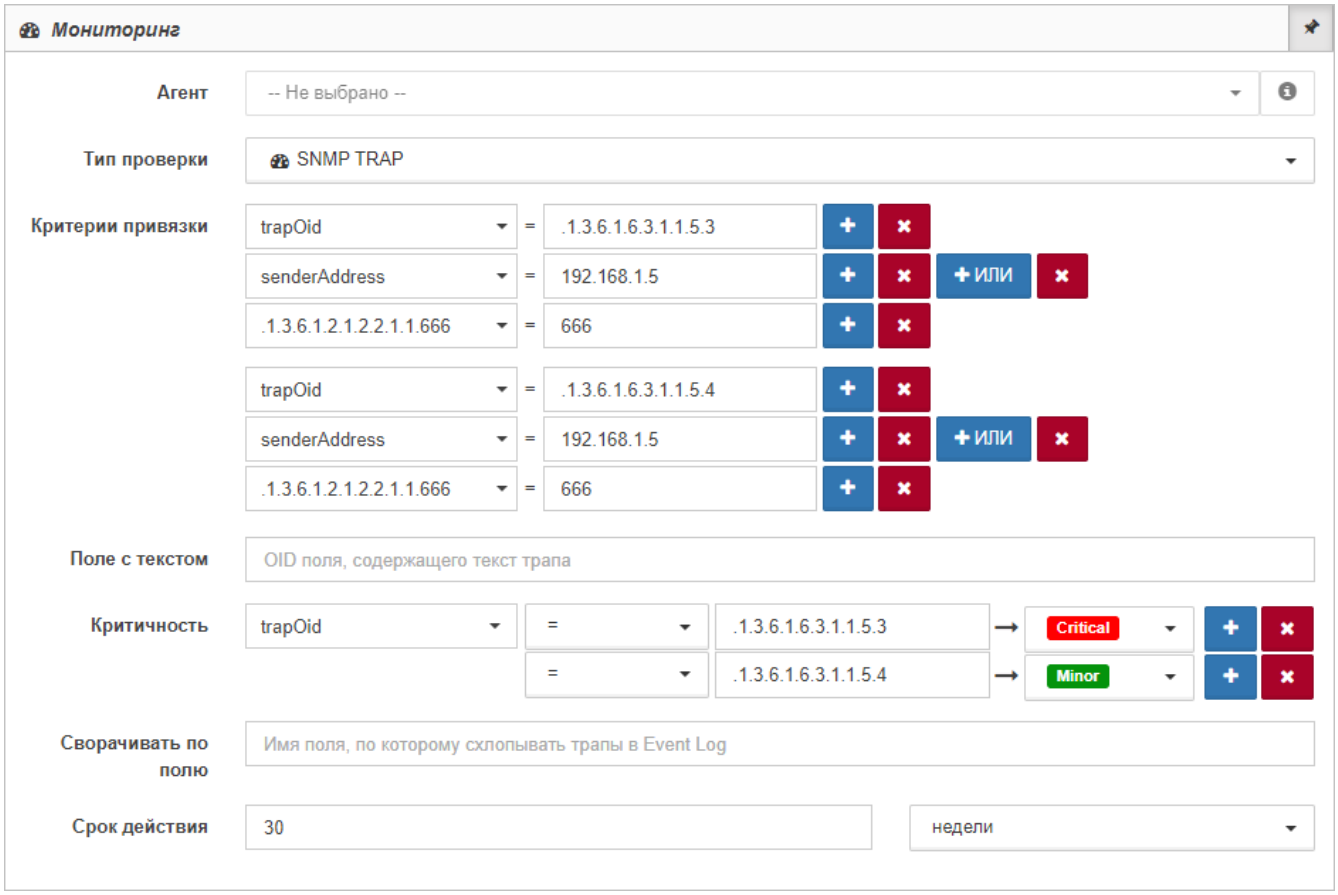
Проверка позволяет "привязать" SNMP Trap'ы к объекту по одному значению или группе значений, например, по значению поля **Trap OID** для SNMP Trap v.2 или по значениям полей **Enterprise OID** и **Agent Address** для SNMP Trap v.1.

NOTE

Структура SNMP Trap v.2: `uptime trapoid [OID TYPE VALUE] …`

Структура SNMP Trap v.1: `enterprise-oid agent-address trap-type(Generic Trap) specific-type(Specific Trap) uptime(Time Stamp) [OID TYPE VALUE]…`

5.3.14.1. Пример настройки



Поле	Описание
Критерии привязки	Ключ и соответствующее ему значение, по которому SNMP Trap сопоставляется с объектом. Кнопка + добавляет критерий через оператор И, позволяя тщательнее фильтровать SNMP Trap'ы. Кнопка + ИЛИ добавляет критерий через оператор ИЛИ, позволяя привязывать несколько SNMP Trap'ов к одному объекту. Кнопка × справа от кнопок добавления критерия удаляет критерий или систему критериев соответственно.
Поле с текстом	OID поля, значение которого будет отображаться в столбце **Текст** журнала событий.
Критичность	Параметр, оператор сравнения, значение параметра и соответствующий уровень критичности, используемые для настройки цветового отображения данных в журнале событий. Кнопка + добавляет набор опций для параметра. Кнопка × удаляет набор опций.
Сворачивать по полю	Дополнительный OID для группировки данных в журнале событий.
Срок действия	Период, в течение которого актуальны полученные данные (по умолчанию 60 секунд).

5.3.14.2. Результат выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:

Данные

senderAddress	bindings..1.3.6.1.2.1.1.3.0	bindings..2.2.2.2	trapOid	text
192.168.1.35	2:35:18.44	Hello1	.1.1.1.1.1	Hello1

Поле	Описание
senderAddress	IP-адрес отправителя.
bindings..1.3.6.1.2.1.1.3.0	Uptime хоста.
bindings..x.x.x.x	Текстовое сообщение из SNMP Trap, отображающееся в столбце **Текст** журнала событий.
trapOid	Значение указанного ключа, по которому SNMP Trap соответствует объекту.
text	Текст полученного SNMP Trap.

TIP

Систему можно настроить так, чтобы SNMP OID в таблице отображались в текстовом (человекочитаемом) формате. Для этого необходимо указать:

- параметр **enableSnmpTranslate: true** в конфигурационном файле клиента,
- сервер для трансляции SNMP OID в текстовый формат в параметре **snmp_mib_url** раздела **Rest_server** конфигурационного файла сервера.

5.3.14.3. Консольная программа

Данная проверка является аналогом программы **Net-SNMP**, в результате работы которой SNMP Trap'ы будут записываться в log-файл.

5.3.14.4. Использование SNMP v3

Начиная с версии 5.0 Агент поддерживает протокол SNMP v3.

Агент поддерживает все три режима безопасности:

- noAuthNoPriv
- authNoPriv с аутентификацией SHA-512
- authPriv с аутентификацией SHA-512 и шифрованием AES-128.

Чтобы включить поддержку SNMPv3 трапов следующие настройки должны быть заданы в файле конфигурации агента:

- agent.snmpTrapV3User,
- agent.snmpTrapV3AuthPassword,
- agent.snmpTrapV3PrivacyPassword.

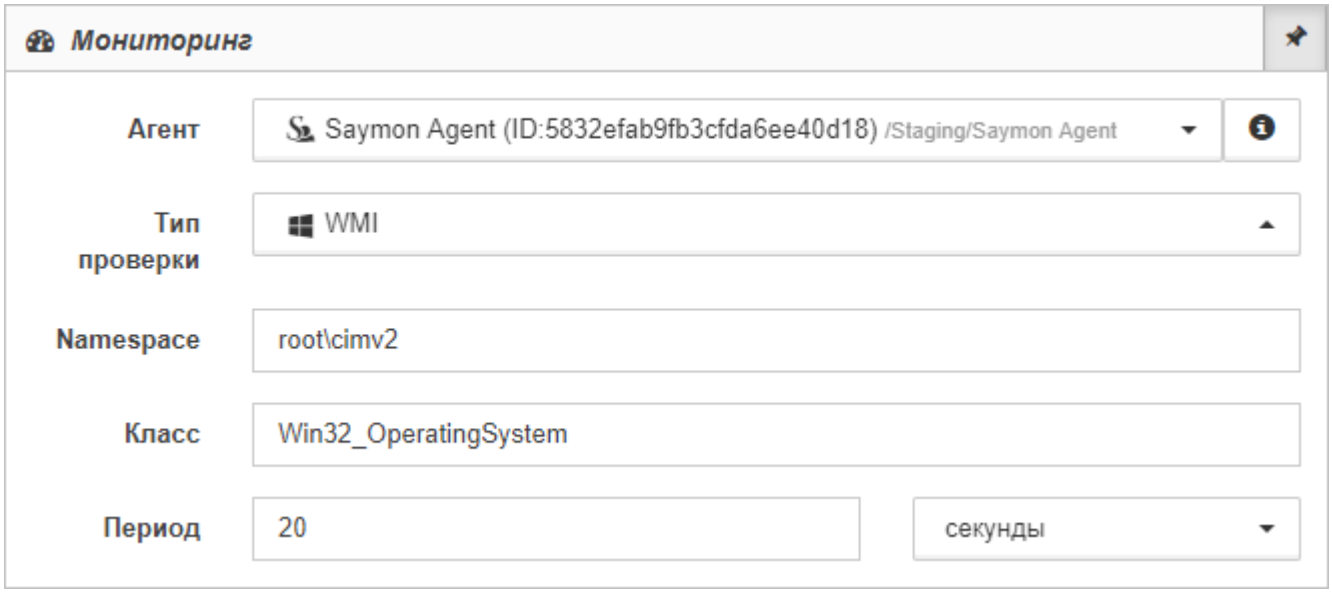
NOTE

Эти параметры не имеют стандартных значений.

5.3.15. WMI-сенсор

Проверка позволяет собирать информацию на операционных системах семейства Windows с помощью технологии "Windows Management Instrumentation" (WMI), получая сведения о классах WMI из указанного пространства имён.

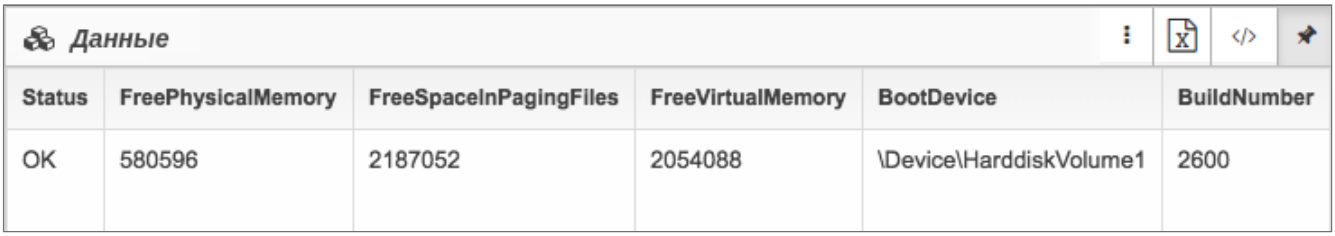
5.3.15.1. Пример настройки



Поле	Описание
Namespace	Используемое пространство имён WMI (по умолчанию **root\cimv2**).
Класс	Класс WMI, свойства которого необходимо получить, например, Win32_OperatingSystem.
Период	Период выполнения проверки (по умолчанию 60 секунд).

5.3.15.2. Результат выполнения

После успешного выполнения проверки в секции Данные появится следующая информация:



Status	FreePhysicalMemory	FreeSpaceInPagingFiles	FreeVirtualMemory	BootDevice	BuildNumber
OK	580596	2187052	2054088	\Device\HarddiskVolume1	2600

Поле	Описание
Поля с названиями свойств запрошенного объекта	Имена свойств запрошенного WMI-объекта и их значения.

5.3.15.3. Консольная программа

Проверка является аналогом командлета **Get-WmiObject**, выполняемого в **PowerShell**:

```
Get-WmiObject -namespace root\cimv2 -class Win32_OperatingSystem | select *


Status                                  : OK
Name                                    : Microsoft Windows XP
Professional|C:\WINDOWS|\Device\Harddisk0\Partition1
FreePhysicalMemory                      : 589436
FreeSpaceInPagingFiles                  : 2189304
FreeVirtualMemory                       : 2054088
__GENUS                                 : 2
__CLASS                                 : Win32_OperatingSystem
__SUPERCLASS                            : CIM_OperatingSystem
__DYNASTY                               : CIM_ManagedSystemElement
__RELPATH                               : Win32_OperatingSystem.Name="Microsoft Windows XP
Professional|C:\\WINDOWS|\
                                          \Device\\Harddisk0\\Partition1"
__PROPERTY_COUNT                        : 61
__DERIVATION                            : {CIM_OperatingSystem, CIM_LogicalElement,
CIM_ManagedSystemElement}
__NAMESPACE                             : root\cimv2
BootDevice                              : \Device\HarddiskVolume1
BuildNumber                             : 2600
BuildType                               : Uniprocessor Free
Caption                                 : Microsoft Windows XP Professional
CodeSet                                 : 1251
CountryCode                             : 7
CreationClassName                       : Win32_OperatingSystem
CSCreationClassName                     : Win32_ComputerSystem
CSDVersion                              : Service Pack 3
...
```

5.3.16. Agent Watchdog

Agent Watchdog – сенсор, предназначенный для мониторинга состояния агента.

Этот сенсор собирает данные, перечисленные в таблице ниже.

Параметр	Описание	Тип	Частота изменения
runtimeName	Название среды выполнения JVM	-	-
runtimeVersion	Версия среды выполнения JVM	-	-
totalMemory	Количество памяти, занятое агентом.	BYTE_QUANTITY	SOMETIMES
freeMemory	Незанятая память, доступная агенту.	BYTE_QUANTITY	SOMETIMES
maxMemory	Ограничение памяти, указанное в конфигурации JVM.	BYTE_QUANTITY	SOMETIMES
usedHeapMemory	Количество использованной памяти в куче.	BYTE_QUANTITY	SOMETIMES
usedNonHeapMemory	Количество использованной памяти вне кучи.	BYTE_QUANTITY	SOMETIMES
processCpuLoad	Нагрузка агента на центральный процессор в процентах.	PERCENTILE	ALWAYS
numberOfThreads	Количество потоков в этом агенте.	-	SOMETIMES
snmpTrapsPerSecond	Количество SNMP трапов, обрабатываемых этим агентом.	-	ALWAYS
tasksNumber	Количество задач, которые выполняет этот агент.	-	SOMETIMES
hostName	Имя хоста.	-	NEVER
networkInterfaces	Список IP адресов, обнаруженных агентом. Содержит адреса IPv4 и IPv6.	-	SOMETIMES

6. Безопасность

6.1. Рекомендации по обеспечению информационной безопасности

6.1.1. Смена пароля по умолчанию

После установки, в web-интерфейсе платформы можно авторизоваться с помощью стандартной учётной записи с максимальными правами:

- логин - `admin`,
- пароль - `saymon`.

WARNING

При первом входе через учётную запись `admin` система потребует смены пароля.

6.1.2. Ограничение доступа к базам данных

Для ограничения доступа к информации, хранящейся в базах данных, рекомендуется настроить в них авторизацию.

С подробными актуальными инструкциями можно ознакомиться в документации производителей:

- MongoDB - https://docs.mongodb.com/guides/server/auth/
- Redis - https://redis.io/topics/security

Учётные записи для доступа к базам данных необходимо добавить в соответствующие секции конфигурационного файла сервера `/etc/saymon/saymon-server.conf`.

В качестве дополнительной опции можно ограничить доступ к cli-утилитам, поставляемым с базами данных:

```
$ sudo chmod 700 /usr/bin/mongo
$ sudo chmod 700 /usr/bin/redis-cli
```

6.1.3. Ограничение доступа к конфигурационным файлам

Конфигурационный файл сервера содержит пароли для подключения к базам данных. Рекомендуется ограничить доступ к этим файлам с помощью команды:

```
$ sudo chmod 700 /etc/saymon/
```

6.1.4. Использование сертификата безопасности SSL

SSL-сертификат представляет собой цифровую подпись, с помощью которой можно обеспечить безопасное шифрованное соединение с платформой.

SSL-сертификат используется для шифрования информации и создания безопасных соединений. С помощью этой технологии сервер может отправлять данные клиентам без риска их перехвата третьей стороной.

Существуют следующие способы получения SSL-сертификата:

- получение бесплатного SSL-сертификата в удостоверяющем центре (например, Let's Encrypt);
- создание самоподписанного SSL-сертификата;
- приобретение коммерческого SSL-сертификата в удостоверяющем центре.

6.1.4.1. Самоподписанные сертификаты

Самоподписанный сертификат — это сертификат, который пользователь создает самостоятельно.

Преимущества самоподписанных SSL-сертификатов:

- возможность создавать сертификаты бесплатно;
- возможность самостоятельно создавать неограниченное количество сертификатов.

Недостатки самоподписанных SSL-сертификатов:

- в браузерах отображается предупреждение безопасности о том, что сертификат не прошел проверку доверенного центра сертификации;
- возможные ошибки в оформлении и отображении сертификата.

WARNING

Самоподписанные SSL-сертификаты не подходят для использования юридическими лицами. Как правило, такие сертификаты используют одиночные пользователи или группы пользователей на внутренних ресурсах для тестирования или защиты некритичных сервисов.

6.1.4.2. Бесплатные сертификаты

Бесплатный сертификат — это сертификат, который можно получить от сертификационного центра (например, Cloudflare или Let's Encrypt) бесплатно.

Преимущества бесплатных SSL-сертификатов:

- возможность получить сертификат бесплатно;
- высокая скорость получения сертификата;
- простота установки сертификата.

Недостатки бесплатных SSL-сертификатов:

- низкий уровень безопасности для пользователей (сертификаты удостоверяют только доменное имя);

- короткий срок действия (3 месяца);
- возможность утраты доверия к сертификату со стороны поисковых систем и браузеров;
- отсутствие печати доверия удостоверяющего центра;
- для установки и обновления сертификата требуются навыки администрирования;
- отсутствие технической поддержки.

TIP

Бесплатные SSL-сертификаты рекомендуется использовать для знакомства с технологией перед приобретением коммерческого SSL-сертификата.

6.1.4.3. Коммерческие сертификаты

Коммерческий сертификат — это сертификат, который можно приобрести в сертификационном центре или у партнёра-распространителя на возмездной основе.

Преимущества коммерческих SSL-сертификатов:

- возможность выбора типа сертификата с нужным уровнем защиты как для физических, так и для юридических лиц;
- высокий уровень безопасности;
- наличие печати доверия;
- большой срок действия (1 год);
- совместимость с большим количеством браузеров;
- отсутствие уведомлений со стороны системы безопасности браузера;
- техническая поддержка по установке и использованию SSL-сертификата.

TIP

Коммерческие SSL-сертификаты рекомендуется использовать для защиты интернет-ресурсов малого, среднего и крупного бизнеса.

6.1.4.4. Регистрация доменного имени для web-сервера

Перед установкой и настройкой сертификата безопасности SSL необходимо зарегистрировать доменное имя (например, example.com). При регистрации доменного имени можно выбрать услугу первичного и/или вторичного DNS. Для этого можно обратиться к любому регистратору доменных имён.

При выборе услуги вторичного DNS потребуется настроить локально первичный DNS-сервер.

NOTE

Требуется получить от провайдера выделенный (фиксированный) IP-адрес.

Настройка первичного DNS-сервера

На DNS-сервере должны быть настроены следующие записи:

- запись A, где `example.com` указывает на публичный IP-адрес вашего сервера;
- запись A, где `www.example.com` указывает на публичный IP-адрес вашего сервера.

Для домена необходимо создать серверный блок (например, */etc/nginx/sites-available/example.com*).

Также необходимо открыть порты для трафика по протоколам HTTP и HTTPS (80 и 443 соответственно).

Пример открытия портов с помощью брандмауэра **ufw**:

```
$ sudo ufw allow 80

$ sudo ufw allow 443
```

6.1.4.5. Получение бесплатного SSL-сертификата от Let's Encrypt

Let's Encrypt — это центр сертификации, с помощью которого можно получать и устанавливать бесплатные сертификаты безопасности SSL.

Процесс получения, установки и последующей настройки сертификата для автоматического продления полностью автоматизирован посредством программного клиента Certbot.

Чтобы получить бесплатный сертификат Let's Encrypt, необходимо выполнить следующие действия:

- установить клиента Certbot;
- настроить конфигурацию Nginx;
- настроить доступ к HTTPS через брандмауэр;
- получить сертификат SSL;
- проверить автоматическое обновление Certbot.

Установка клиента Certbot

Для установки клиента Certbot необходимо выполнить следующую команду:

```
$ sudo apt install certbot python3-certbot-nginx
```

После установки Certbot готов к использованию.

После этого необходимо настроить конфигурацию Nginx.

Настройка конфигурации Nginx

Для автоматической настройки SSL Certbot необходимо настроить серверный блок в конфигурации Nginx. Для этого нужно указать директиву `server_name`, которая соответствует домену, для которого запрашивается сертификат.

Откройте файл конфигурации вашего домена в текстовом редакторе (например, `nano`):

```
$ sudo nano /etc/nginx/sites-available/example.com
```

В строке `server_name` укажите зарегистрированное доменное имя, например:

```
server_name example.com www.example.com;
```

Сохраните изменения. После этого закройте редактор и проверьте синтаксис внесенных правок:

```
$ sudo nginx -t
```

TIP

Если отображается сообщение об ошибке, необходимо открыть файл серверного блока и проверить его на наличие ошибок или опечаток.

Для завершения необходимо перезагрузить Nginx для загрузки новой конфигурации:

```
$ sudo systemctl reload nginx
```

После этого Certbot сможет найти серверный блок и автоматически его обновлять.

Настройка доступа к HTTPS через брандмауэр

Чтобы разрешить HTTPS-трафик на web-сервере, необходимо настроить брандмауэр.

Пример команды для просмотра текущих настроек брандмауэра **ufw**:

```
$ sudo ufw status
```

Если на web-сервере разрешен только HTTP-трафик, то результат будет выглядеть следующим образом:

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx HTTP                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Nginx HTTP (v6)            ALLOW       Anywhere (v6)
```

Затем необходимо активировать профиль Nginx Full и удалить профиль Nginx HTTP.

Пример команды для брандмауэра **ufw**:

```
$ sudo ufw allow 'Nginx Full'

$ sudo ufw delete allow 'Nginx HTTP'
```

Результат должен выглядеть следующим образом:

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx Full                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Nginx Full (v6)            ALLOW       Anywhere (v6)
```

Получение сертификата SSL

Запустите Certbot с параметром `--nginx`, используя опцию `-d`, чтобы указать доменное имя, для которого нужно использовать сертификат, например:

```
$ sudo certbot --nginx -d example.com
```

Если Certbot запускается первый раз, то будет предложено указать адрес электронной почты и принять условия обслуживания. После этого Certbot свяжется с сервером Let's Encrypt и проверит принадлежность домена, для которого запрашивается сертификат.

После этого Certbot запросит желаемый вариант настройки HTTPS:

```
Output
Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel):
```

После выбора желаемого варианта конфигурация будет обновлена. Nginx перезагрузится для вступления новых настроек в силу.

После завершения процесса Certbot отобразит сообщение, указывающее место хранения сертификатов:

```
Output
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/example.com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/example.com/privkey.pem
   Your cert will expire on 2020-08-18. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
   with the "certonly" option. To non-interactively renew *all* of
   your certificates, run "certbot renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

По завершению этого процесса сертификаты загружены, установлены и активированы.

Проверка автоматического обновления Certbot

Сертификаты Let's Encrypt действуют в течение 90 дней. Клиент Certbot добавляет таймер `systemd`, который запускается два раза в день и автоматически продлевает все сертификаты, срок действия

которых составляет менее 30 дней.

Для проверки статуса таймера необходимо выполнить следующую команду:

```
$ sudo systemctl status certbot.timer
```

```
Output
● certbot.timer - Run certbot twice daily
     Loaded: loaded (/lib/systemd/system/certbot.timer; enabled; vendor preset: enabled)
     Active: active (waiting) since Mon 2020-05-04 20:04:36 UTC; 2 weeks 1 days ago
    Trigger: Thu 2020-05-21 05:22:32 UTC; 9h left
   Triggers: ● certbot.service
```

Чтобы проверить процесс обновления, нужно выполнить следующую команду:

```
$ sudo certbot renew --dry-run
```

NOTE

Certbot будет автоматически продлевать сертификаты и перезагружать Nginx, чтобы изменения вступали в силу. Если процесс автоматического обновления не будет выполнен, то на указанный вами адрес электронной почты поступит сообщение с предупреждением о том, что срок действия сертификата подходит к концу.

6.1.4.6. Создание самоподписанного SSL-сертификата

NOTE

Поскольку самоподписанный сертификат не подписан ни одним из доверенных центров сертификации, включенных в браузеры, то его нельзя использовать для проверки подлинности сервера.

Для создания самоподписанного SSL-сертификата необходимо выполнить следующие шаги:

1. Создать SSL-сертификат.
2. Настроить Nginx для использования самоподписанного SSL-сертификата.
3. Установить самоподписанный сертификат.

Создание SSL-сертификата

Для работы TLS/SSL необходимо создать пару открытого сертификата и закрытого ключа. Закрытый ключ хранится на сервере и шифрует данные, отправляемые клиентам, а SSL-сертификат доступен всем пользователям, которые подключаются к серверу. С помощью этого сертификата можно расшифровать содержимое, подписанное соответствующим SSL-ключом.

Для создания файлов сертификата нужно использовать следующую команду:

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
```

NOTE

При создании SSL-сертификата также создается файл ключа, который позволяет серверу безопасно обрабатывать шифрованные данные.

где:

- `openssl`: это инструмент командной строки, предназначенный для создания сертификатов, ключей и других файлов OpenSSL и управления ими;
- `req -x509`: задает управление запросами на подписание сертификатов X.509 (CSR);
- `-nodes`: предписывает пропустить опцию защиты сертификата кодовой фразой после перезапуска сервера;
- `-days 365`: устанавливает период действия сертификата сроком на 1 год. Многие браузеры отклоняют любые сертификаты, срок действия которых превышает этот период;
- `-newkey rsa:2048`: указывает, что требуется сгенерировать новый сертификат и новый ключ одновременно; часть `rsa:2048` предписывает создать ключ RSA длиной 2048 бит;
- `-keyout`: указывает, где разместить генерируемый файл закрытого ключа;
- `-out`: указывает, где разместить создаваемый сертификат.

После ввода команды отображается список диалогов, в которых нужно ввести необходимую информацию.

В строке `Common Name` нужно ввести имя хоста или публичный IP-адрес, который будет использоваться для доступа к серверу. Указываемое значение должно совпадать с данными, вводимыми в адресную

строку браузера.

Полный список диалогов выглядит следующим образом:

```
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:Example
Locality Name (eg, city) [Default City]:Example
Organization Name (eg, company) [Default Company Ltd]:Example Inc
Organizational Unit Name (eg, section) []:Example Dept
Common Name (eg, your name or your server's hostname) []:your_domain_or_ip
Email Address []:admin@example.com
```

Созданные файлы сохраняются в соответствующие подкаталоги в каталоге */etc/ssl*. Подкаталоги задаются с помощью параметров `-keyout` и `-out`.

Чтобы создать ключ для поддержки PFS, используйте следующую команду:

```
$ sudo openssl dhparam -out /etc/nginx/dhparam.pem 4096
```

Этот процесс занимает несколько минут. Созданный ключ будет помещен в каталог */etc/nginx*.

Настройка Nginx для использования самоподписанного SSL-сертификата

После создания ключа и сертификата нужно отредактировать настройки сервера Nginx. Для этого нужно выполнить следующее:

1. Указать расположение файлов ключа и SSL-сертификата.
2. Задать настройки SSL.
3. Настроить Nginx для поддержки SSL.
4. Настроить брандмауэр.
5. Сохранить изменения в конфигурации Nginx.
6. Проверить шифрование.

Задание расположения файлов ключа и SSL-сертификата

Создайте новый сниппет Nginx в каталоге /etc/nginx/snippets:

```
$ sudo nano /etc/nginx/snippets/self-signed.conf
```

В созданный файл нужно добавить директиву `ssl_certificate`, которая указывает путь к сертификату, и директиву `ssl_certificate_key`, которая указывает путь к закрытому ключу:

```
ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
```

Сохраните изменения и закройте файл.

Задание настроек SSL

Создайте другой сниппет конфигурации Nginx, в котором будут храниться настройки SSL, например:

```
$ sudo nano /etc/nginx/snippets/ssl-params.conf
```

Скопируйте следующий код в файл ssl-params.conf:

```
ssl_protocols TLSv1.2;
ssl_prefer_server_ciphers on;
ssl_dhparam /etc/nginx/dhparam.pem;
ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384;
ssl_ecdh_curve secp384r1; # Requires nginx >= 1.1.0
ssl_session_timeout  10m;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off; # Requires nginx >= 1.5.9
ssl_stapling on; # Requires nginx >= 1.3.7
ssl_stapling_verify on; # Requires nginx => 1.3.7
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
# Disable strict transport security for now. You can uncomment the following
# line if you understand the implications.
# add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload";
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
```

Сохраните изменения и закройте файл.

Настройка Nginx для поддержки SSL

Создайте резервную копию текущего файла конфигурации Nginx, например:

```
$ sudo cp /etc/nginx/sites/your_domain /etc/nginx/sites/your_domain.bak
```

Откройте созданный файл, чтобы внести в него изменения, например:

```
$ sudo nano /etc/nginx/sites/your_domain
```

В файле нужно обновить директивы listen и включить сниппеты SSL. Этот блок нужно настроить для поддержки трафика SSL по порту 443, а затем создать новый блок server для обработки трафика на порту 80 и автоматической переадресации трафика на порт 443.

В текущем файле обновите директивы listen для поддержки порта 443 и ssl, а затем включите два файла сниппетов: self-signed.conf и ssl-params.conf.

```
server {
    listen 443 ssl;
    listen [::]:443 ssl;
    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;

root /var/www/your_domain/html;
        index index.html index.htm index.nginx-debian.html;

  server_name your_domain.com www.your_domain.com;

  location / {
                try_files $uri $uri/ =404;
        }
}
```

Добавьте второй блок server (после закрывающей фигурной скобки (}) первого блока):

```
server {
    listen 80;
    listen [::]:80;

    server_name your_domain.com www.your_domain.com;

    return 302 https://$server_name$request_uri;
}
```

Это позволит прослушивать порт 80 и выполнять перенаправление на HTTPS.

Сохраните изменения и закройте файл.

Настройка брандмауэра

Если вы используете брандмауэр (например, ufw), то нужно изменить настройки, чтобы разрешить SSL-трафик.

При установке Nginx регистрирует в ufw несколько профилей. Посмотреть доступные профили можно с помощью следующей команды:

```
$ sudo ufw app list
```

Должен отобразиться следующий вывод:

```
Available applications:
  Nginx Full
  Nginx HTTP
  Nginx HTTPS
  OpenSSH
```

Чтобы проверить текущие настройки, используйте следующую команду:

```
$ sudo ufw status
```

Скорее всего, веб-серверу разрешен только HTTP-трафик. Должен отобразиться вывод, как показано ниже:

```
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx HTTP                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Nginx HTTP (v6)            ALLOW       Anywhere (v6)
```

Чтобы разблокировать HTTPS-трафик, можно обновить разрешения для профиля **Nginx Full**, после чего удалить профиль **Nginx HTTP**:

```
$ sudo ufw allow 'Nginx Full'
$ sudo ufw delete allow 'Nginx HTTP'
```

Выполните команду `sudo ufw status` и проверьте, что настройка брандмауэра была выполнена успешно:

```
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx Full                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Nginx Full (v6)            ALLOW       Anywhere (v6)
```

Сохранение изменений в конфигурации Nginx

После настройки брандмауэра нужно перезапустить Nginx, чтобы изменения вступили в силу.

Выполните следующую команду, чтобы убедиться, что в файлах нет синтаксических ошибок:

```
$ sudo nginx -t
```

Если ошибки отсутствуют, должен отобразиться следующий вывод:

```
nginx: [warn] "ssl_stapling" ignored, issuer certificate not found for certificate
"/etc/ssl/certs/nginx-selfsigned.crt"
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

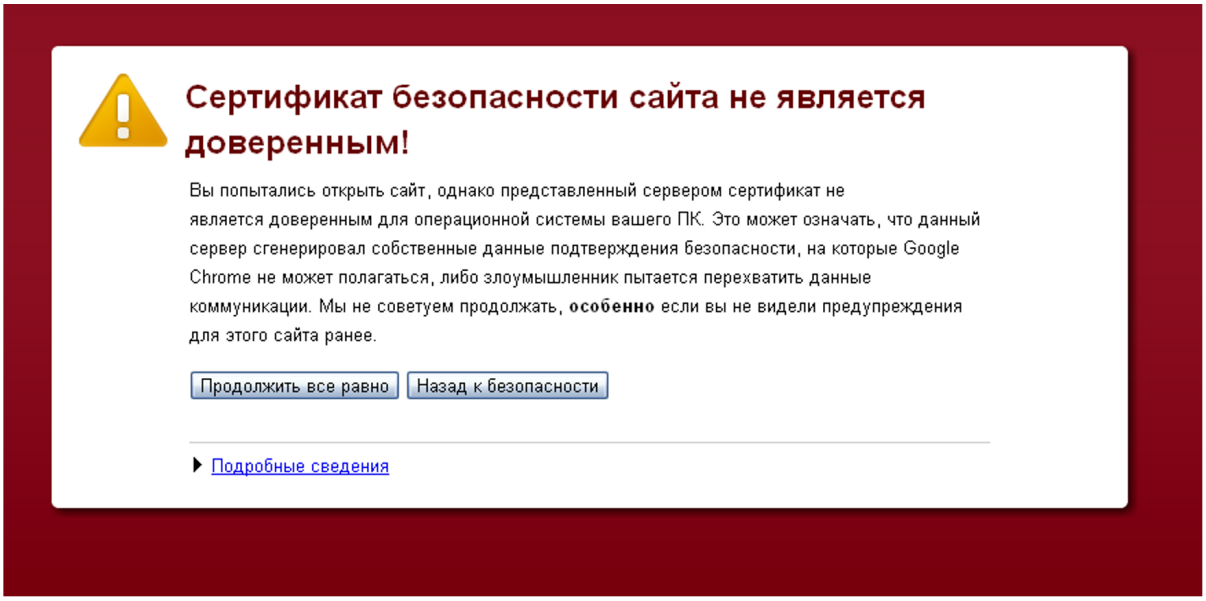
После этого можно перезапустить Nginx для сохранения изменений:

```
$ sudo systemctl restart nginx
```

Проверка шифрования

Подключитесь к web-серверу системы Центральный Пульт, добавив в адресной строке браузера префикс `https://`.

Самоподписанный сертификат не подписан доверенным центром сертификации, который входит в базовый комплект поставки браузеров и операционных систем, поэтому при попытке подключиться к web-серверу системы Центральный Пульт в браузере будет отображаться предупреждение системы безопасности.



Для загрузки страницы необходимо нажать кнопку «Продолжить все равно» (или аналогичную, например: «Дополнительно» или «Продолжить»).

TIP

Если браузер не подключается к серверу, убедитесь, что брандмауэр не блокирует соединение.

Установка самоподписанных сертификатов

Чтобы предупреждение системы безопасности не отображалось, нужно добавить самоподписанный сертификат в системное хранилище доверенных сертификатов или хранилище сертификатов браузера.

Для этого выполните указанные ниже действия.

Скопируйте сертификат в отдельную папку (например: /usr/local/share/ca-certificates):

```
$ cp ./ca.crt /usr/local/share/ca-certificates/losstca.crt
```

После этого необходимо выполнить следующую команду:

```
$ sudo update-ca-certificates
```

Проверить, что система воспринимает сертификат как доверенный, можно с помощью следующей команды:

```
$ curl -I https://localhost
```

TIP

Некоторые браузеры (например, Firefox и Google Chrome) имеют собственные хранилища сертификатов и не используют хранилище системы. Поэтому в каждый такой браузер нужно импортировать самоподписанный сертификат.

Чтобы импортировать самоподписанный сертификат в хранилище сертификатов браузера, выполните указанные ниже действия.

Импорт сертификата с помощью графического интерфейса в Google Chrome

- В настройках браузера нужно перейти в раздел **Конфиденциальность и безопасность** → **Безопасность** → **Управление сертификатами устройства**.
- В открывшемся окне нужно открыть вкладку **Доверенные корневые центры сертификации**.
- Нажать кнопку **Импорт** и выбрать файл корневого сертификата.
- Указать желаемые параметры доверия.

Сертификат будет импортирован в хранилище сертификатов браузера.

Импорт сертификата в терминале с помощью утилиты **certutil**

Для работы с сертификатами требуется пакет libnss3-tools. Для установки пакета используйте следующую команду:

```
$ sudo apt install libnss3-tools
```

База данных сертификатов Google Chrome находится в папке *~/.pki/nssdb*. Чтобы посмотреть доступные сертификаты, можно использовать следующую команду:

```
$ certutil -d ~/.pki/nssdb -L
```

Синтаксис команды для добавления сертификата следующий:

```
$ certutil -d путь/к/базе/данных -A -t "настройки_доверия" -n "имя" -i "/путь/к/файлу"
```

Существует три группы атрибутов доверия:

- для SSL;
- для Email;
- для программного обеспечения и других объектов.

Каждая из групп может содержать следующие атрибуты:

- p — валидный пир;
- P — доверенный пир;
- c — валидный центр сертификации;
- C — доверенный центр сертификации;
- T — доверенный центр сертификации для авторизации клиентов.

Для SSL-сертификатов будет достаточно такой последовательности "TC,,". Полная команда для импорта ca.crt будет выглядеть следующим образом:

```
$ certutil -d sql:~/.pki/nssdb -A -t "TC,," -n "Losst CA" -i ./ca.crt
```

Для просмотра списка сертификатов можно использовать следующую команду:

```
$ certutil -d sql:~/.pki/nssdb -L
```

6.1.4.7. Получение коммерческого сертификата

Коммерческий SSL-сертификат можно приобрести в различных сертификационных центрах (например, рег.ру, FirstSSL, LeaderSSL, Reddock).

Процедура установки таких сертификатов полностью автоматизирована.

Настроить ПО для использования сертификатов помогает техническая поддержка.

6.1.5. Шифрование паролей

Центральный Пульт поддерживает шифрование паролей и другой конфиденциальной информации, содержащейся в конфигурационных файлах и настройках проверок.

6.1.5.1. Формат зашифрованных параметров

Зашифрованные значения параметров указываются в следующем формате:

```
<tags[0]><prefix>:<salt>:<encrypted_value><tags[1]>
```

Например:

```
"value": "<<crypt:db7676ec4c7624788349b75e43a3b6a4:1a67b106fabeadf2ac8458395c5e644d>>"
```

Если значение поля окружено тегами, но у этого поля нет префикса, то сервер зашифрует это поле при перезапуске.

Если у значения поля есть и теги и префикс, то сервер будет работать с этим полем как с зашифрованым.

Если значение не имеет тегов и префикса, то Центральный Пульт предполагает, что для этого поля шифрование не используется.

Теги и префикс, окружающие зашифрованное поле, можно настроить в Конфигурации сервера.

6.1.5.2. Шифрование файлов конфигурации

Шифрование включается в конфигурации сервера с помощью параметра `server_config`:

```
"encrypt": {
  "server_config": true
}
```

Поля конфигурации, которые нужно зашифровать, помечаются тегами (по умолчанию – `<<` и `>>`):

```
{
  "cache": {
    "password": "testPass1"
  },
  "fieldToEncrypt": "<<testPass2>>",
  "openTsdb": {
    "host": "localhost",
    "port": 4242
  }
}
```

После перезапуска сервера, Центральный Пульт зашифрует все помеченные поля, а также все поля с именем `pass` и `password`:

```
{
  "cache": {
    "password": "<<crypt:bd5ea51c5ee453220a2616645672e96b:1a67b106fabeadf2ac8458395c5e644d>>"
  },
  "fieldToEncrypt": "
<<crypt:db7676ec4c7624788349b75e43a3b6a4:5602ae7b1a4a8ae8ea6cca845f5d4c5f>>",
  "openTsdb": {
    "host": "localhost",
    "port": 4242
  }
}
```

TIP

Теги и префикс, окружающие зашифрованное поле, можно настроить в Конфигурации сервера.

6.1.5.3. Шифрование параметров проверок

Некоторые параметры проверок могут содержать конфиденциальную информацию (например, пароли). Центральный Пульт поддерживает шифрование таких параметров.

Для включения шифрования параметров необходимо включить параметр `encrypt.db` в Конфигурации сервера:

```
"encrypt": {
  "db": true
}
```

Шифрование параметров поддерживается в следующих проверках:

- monitoring:sensors/database-query.html (параметр JdbcPassword)
- monitoring:sensors/jmx-probe.html (параметр JmxPassword)
- monitoring:sensors/ftp-probe.html (параметр FtpPassword)
- monitoring:sensors/ssh.html (шифрование закрытого ключа)

После установки этого свойства значения приведённых выше полей будут сохраняться в базе в зашифрованном виде.

При получении информации о проверке через REST запросы зашифрованные параметры будут заменяться 4 звездочками (`****`).

NOTE

При возникновении ошибки при расшифровке или при отключении шифрования после зашифровки параметра сервер вернёт пустую строку вместо значения зашифрованных полей.

6.1.6. Настройка Comet- и Rest-сервера для использования сертификата безопасности SSL

6.1.6.1. Настройка Comet-сервера

Настройка Comet-сервера для использования SSL-сертификата осуществляется при помощи файла конфигурации сервера `/etc/saymon/saymon-server.conf`.

Для этого в секции **Server** нужно задать значения для следующих параметров:

- **comet_ssl_certificate** - путь к сертификату (относительно каталога установки сервера).
- **comet_ssl_key** - путь к ключу (относительно каталога установки сервера).

6.1.6.2. Настройка Rest-сервера

Все запросы из графического интерфейса web-приложения проксируются в Rest-сервер через Nginx. Для шифрования передаваемых данных нужно настроить Nginx для использования SSL-сертификата (подробнее см. Создание самоподписанного SSL-сертификата).

6.1.7. Настройка Kafka для использования сертификата безопасности SSL

С помощью шифрования SSL можно обеспечить безопасное подключение агента к серверу Kafka.

Для настройки шифрования используется файл конфигурации сервера `/etc/saymon/saymon-server.conf` и файл конфигурации агента `<папка_установки агента>/conf/agent.properties`.

В файле конфигурации сервера в секции **Kafka** нужно задать значения для следующих параметров:

- **kafka.ssl** - пути к файлам ключа и сертификата в следующем формате:

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

Для применения изменений необходимо перезапустить службу saymon-server:

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

В файле конфигурации агента нужно задать значения для следующих параметров:

- **server.secure** - подключение агента к серверу Kafka в режиме шифрования TLS:

  `server.secure=true`

- **kafka.security.protocol** - протокол, используемый для взаимодействия с брокерами:

  `kafka.security.protocol=SSL`

При использовании SSl-сертификатов публичных удостоверяющих центров:

- **kafka.ssl.truststore.location** - расположение файла ключа в truststore.
- **kafka.ssl.truststore.password** - пароль для файла truststore. Если пароль не задан, сконфигурированный файл также будет использоваться, но без проверки целостности.

При использовании самоподписных SSL-сертификатов:

- **server.ca.certificate** - значение PEM-сертификата в текстовом виде.
- **server.ca.location** - путь к файлу в PEM формате.

6.1.8. Настройка защищенного подключения через протокол MQTT

Ниже описана процедура настройки защищенного соединения между брокером Mosquitto и клиентом по протоколу MQTT. В качестве клиентов могут выступать любые устройства с библиотекой MQTT.

Чтобы настроить SSL-шифрование на стороне брокера, нужно внести в файл конфигурации брокера **mosquitto.conf** (по умолчанию `/etc/mosquitto/mosquitto.conf`) указанные ниже изменения:

- Откройте файл конфигурации с помощью следующей команды:

```
$ sudo nano /etc/mosquitto/conf.d/mosquitto.conf
```

- Добавьте в файл конфигурации следующую секцию параметров:

```
. . .
listener 1883 localhost

listener 8883
certfile /etc/encryption/cert.pem
cafile /etc/ encryption /chain.pem
keyfile /etc/ encryption /privkey.pem
```

- Убедитесь, что в конце файла осталась пустая строка.
- Закройте файл и сохраните изменения.
- Перезапустите Mosquitto, чтобы обновить конфигурацию:

```
$ sudo systemctl restart mosquitto
```

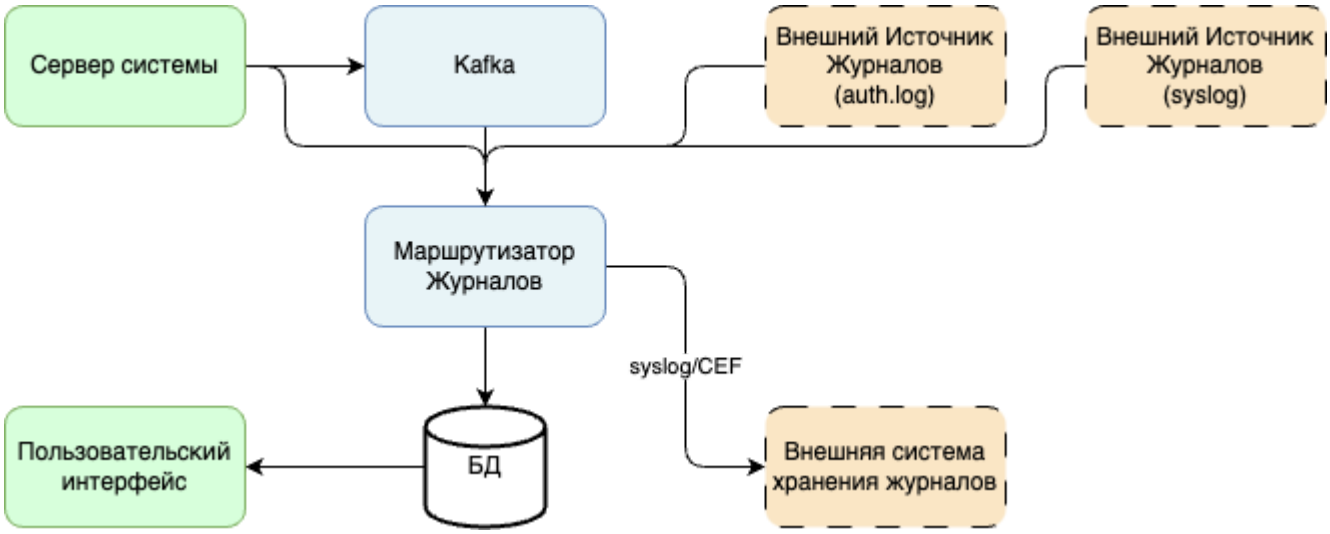
- Обновите конфигурацию брандмауэра, чтобы разрешить подключения для порта 8883:

```
$ sudo ufw allow 8883
```

6.1.9. Журнал событий безопасности

Центральный Пульт поддерживает журналирование событий, произошедших в системе, для последующего аудита администратором или службой безопасности.

6.1.9.1. Архитектура журналирования событий безопасности



Архитектура состоит из следующих элементов:

- Источники журналов
- Маршрутизатор журналов (log-shipper)
- Системы хранения и обработки журналов

Источники журналов

В роли источников журналов выступают:

- тематические каналы журналирования сервера системы Центральный Пульт, отправляющие записи журнала во внешние системы по протоколу UDP или в топики Kafka.
- внешние системы:
  - топики Kafka,
  - локальные файлы журналов (например syslog или auth.log),
  - любые другие системы передающие или хранящие сообщения.

Маршрутизатор журналов (log-shipper)

Маршрутизатор журналов (log-shipper) является отдельным самостоятельным компонентом, основной задачей которого является:

- принимать записи журналов из разных источников,
- производить фильтрацию полученных сообщений,
- преобразовывать полученные сообщений из одного формата в другой,
- отправлять преобразованные данные в различные системы хранения и обработки.

Центральный Пульт использует маршрутизатор fluentd.

Системы хранения и обработки журналов

Маршрутизатор отправляет полученные записи журналов в БД сервера системы Центральный Пульт, откуда их можно получить через пользовательский интерфейс или REST API.

Также можно настроить отправку журналов во внешние системы хранения по заданному в маршрутизаторе алгоритму. Маршрутизатор может отправлять в системы только определённые сообщения, либо отправлять одинаковые сообщения в несколько систем.

NOTE

Количество внешних систем хранения не ограничено.

6.1.9.2. Включение журнала безопасности

Включение отправки сообщений

Для отправки событий ИБ необходимо:

1. В конфигурационном файле системы журналирования `/etc/saymon/logger.json` добавить секцию `channels`. Если этого файла нет, его следует создать.

TIP

Все возможные каналы и их настройки описаны в статье server-configuration:config/log-config.html.

```
{
    "categories": {
        "Default": "Warn"
    },
    "channels": {
        "session": [
            {
                "type": "kafka",
                "options": {
                    "server": "localhost:9092",
                    "topic": "LOG_SESSION"
                }
            }
        ],
        "audit": [
            {
                "type": "udp",
                "options": {
                    "server": "localhost:10001"
                }
            }
        ],
        "faults": [
            {
                "type": "udp",
                "options": {
                    "server": "localhost:10001"
                }
            }
        ]
    }
}
```

NOTE

Каждый канал (`session`, `audit` или `faults`) можно настроить как на отправку в Kafka (в свой топик: `LOG_SESSION`, `LOG_AUDIT`, `LOG_FAULTS`), так и на отправку во внешние системы (например syslog) по UDP.

2. Запустить "маршрутизатор" `fluentd`.

   Для этого нужно выполнить следующие действия:

   a. На сервере перейти в каталог `/etc/saymon/`:

```
$ cd /etc/saymon/
```

b. Скачать конфигурационные файлы fluentd:

```
$ sudo wget {site-url}/_attachments/enable-security-log/fluentd.tar.gz
```

c. Распаковать архив `/etc/saymon/fluentd/` :

```
$ tar -xvzf fluentd.tar.gz
```

d. Выполнить следующую команду:

```
$ sudo docker run -d --restart=unless-stopped --name=fluentd --network=host -v
/etc/saymon/fluentd/config:/fluentd/etc -v /var/log/auth.log:/var/log/auth.log:ro -
uroot --log-opt max-file=3 --log-opt max-size=10m portus.saymon.info/saymon-
public/fluentd:latest
```

3. Запустить сервер системы Центральный Пульт

После этого все сообщения ИБ от сервера системы Центральный Пульт будут отправляться в "маршрутизатор" `fluentd`

Включение интерфейса журнала безопасности

Для просмотра записей журнала ИБ необходимо:

1. В конфигурационном файле сервера `/etc/saymon/saymon-server.conf` добавить секцию `logdb` :

```
{
  "server": { ... },
  "logdb": {
    "url": "mongodb://localhost:27017/logs",
    "collection": "securityLogs"
  }
}
```

2. Перезапустить сервер системы Центральный Пульт.
3. Дать права на просмотр журнала событий ИБ. Для этого в меню Права на операции необходимо включить параметр «Мониторинг логов и событий ИБ».

После назначения пользователю соответствующих прав, в основном меню ему станет доступна ссылка на «Журнал событий ИБ». По этой кнопке можно перейти в интерфейс журнала безопасности.

6.1.9.3. События журнала безопасности

События сессии

Название События	ID Класса События	Описание	Критичность
`sessionOpen`	'AAA'	'Session is opened'	3
`sessionClose`	'AAC'	'Session is closed'	3
`sessionExpire`	'AAE'	'Session expired'	6
`loggedWithKey`	'AAK'	'Logged with key'	3

События авторизации

Название События	ID Класса События	Описание	Критичность
`authorizationError`	'AAF'	'Authorization error'	8

События аудита

Название События	ID Класса События	Описание	Критичность
`auditEvent`	'AUDIT'	'Changing in the system'	0

Объекты

Название События	ID Класса События	Описание	Критичность
`entityAdd`	'OCA'	'Adding an object'	3
`entityRemove`	'OCD'	'Deleting an object'	8
`entityChange`	'OCH'	'Changing an object'	3

Пользователи

Название События	ID Класса События	Описание	Критичность	Примечания
`userActivation`	'CUA'	'User activation'	3	
`userDeactivation`	'CUP'	'User deactivation'	3	
`userExpire`	'CUE'	'User lifetime expiration'	3	
`userRemove`	'CUD'	'Deleting a user'	8	
`userChangePermissions`	'CPU'	'Change user permission'	3	Генерируется при изменении свойств пользователя `permissions` и `objectPermissions`.

Пароли

Название События	ID Класса События	Описание	Критичность
`passwordExpire`	'PAE'	'Password lifetime expiration'	3
`passwordResetByAdmin`	'PAR'	'Reset user password by admin'	3
`passwordChangeByUser`	'PCH'	'Password change by user'	3

Данные

Название События	ID Класса События	Описание	Критичность
`dataDownload`	'WDN'	'Data download'	6
`dataUpload`	'WUP'	'Data upload'	6

6.1.9.4. Поддержка формата CEF

Центральный Пульт поддерживает отправку сообщений ИБ в формате CEF.

При отправке сообщений в таком формате, сообщение имеет следующий вид:

```
CEF:Version|Device Vendor|Device Product|Device Version|Signature ID|Name|Severity|Extension
```

, где:

- `Version` — версия формата CEF. Центральный Пульт использует версию `0`.
- `Device Vendor` – ROSSINNO
- `Device Product` – SAYMON
- `Device Version` – версия системы Центральный Пульт.
- `Signature ID` — уникальный идентификатор типа события.
- `Name` — название события.
- `Severity` — критичность события.
- `Extension` - дополнительная информация о событии.

Поля Extension

В поле Extension передаются следующие поля:

Идентификационные
- `dproc` – текст `saymon-server` | `auth.log`
- `suser` – субъект действия
- `src` – IP-адрес субъекта
- `shost` – DNS-имя субъекта
- `dst` – IP-адрес saymon-server
- `dhost` – DNS-имя saymon-server

Результат действия
- `outcome` - текст `Failure` | `Success`
- `msg` - подробная информация по деталям действия в формате JSON

Опциональные поля
В таблице перечислены опциональные поля для конкретных событий классов.

Поле	Описание	События
cn1	Идентификатор сессии	AAA,AAC,AAF,AAK,AAE,AAO
cn3	Число 3	AAA,AAC,AAF,AAK,AAE,AAO
fname	Идентификатор объекта	OCA,OCD,OCH
duser	Логин пользователя, над которым производится действие	CUA,CUP,CUD,PAR
reason	Причина ошибки	События из канала faults
cs1	Текст пароля	PCH,PAR

Примеры

Примеры сообщений в формате CEF.

Выход из терминала на сервере системы Центральный Пульт:

```
CEF:0|ROSSINNO|SAYMON|3.12.86|AA-CLI|CRON|6|dproc=auth.log end=1709205421000
msg="pam_unix(cron:session): session closed for user root"
```

Вход через веб-интерфейс системы Центральный Пульт:

```
CEF:0|ROSSINNO|SAYMON|3.12.86|AAA|Session is opened|3|dproc=saymon-server end=1709194963050
outcome=Success suser= src=10.0.2.2 shost=_gateway dst=127.0.0.1 dhost=127.0.0.2
cn1=23e57682f98e0cebe1b1979e6c0b71a9 cn3=3  msg=
{"startTime":1709194963011,"expiredAt":1709198563011,"userId":"5e4cd4668f7a9c6f9a128e16","logi
n":"admin","sessionId":"23e57682f98e0cebe1b1979e6c0b71a9"}
```

Добавление нового объекта:

```
CEF:0|ROSSINNO|SAYMON|3.12.86|OCA|Adding an object|3|dproc=saymon-server end=1709194977890
outcome=Success suser=admin src=10.0.2.2 shost=_gateway dst=127.0.0.1 dhost=127.0.0.2
fname=65e03ee10b298625a3a9fae1  msg=
{"entityType":"Object","entityId":"65e03ee10b298625a3a9fae1","kind":"Object"}
```

Удаление объекта:

```
CEF:0|ROSSINNO|SAYMON|3.12.86|OCD|Deleting an object|8|dproc=saymon-server end=1709195003861
outcome=Success suser=admin src=10.0.2.2 shost=_gateway dst=127.0.0.1 dhost=127.0.0.2
fname=65e03ee10b298625a3a9fae1  msg=
{"entityType":"Object","entityId":"65e03ee10b298625a3a9fae1","kind":"Object"}
```

Изменение прав пользователя:

```
CEF:0|ROSSINNO|SAYMON|3.12.86|CPU|Change user permissions|3|dproc=saymon-server
end=1709195023216 outcome=Success suser=admin src=10.0.2.2 shost=_gateway dst=127.0.0.1
dhost=127.0.0.2  msg={"entityType":"User","entityId":"65a527d3679b6d24f8fce0d1","changes":
{"permissions":["read-security-log"]},"action":"user props changing","details":
{"login":"dgu_user"}}
```

Сброс пароля пользователя администратором:

```
CEF:0|ROSSINNO|SAYMON|3.12.86|PAR|Reset user password by admin|3|dproc=saymon-server
end=1709195040369 outcome=Success suser=admin src=10.0.2.2 shost=_gateway dst=127.0.0.1
dhost=127.0.0.2 duser=dgu_user cs1=Password  msg=
{"entityType":"User","entityId":"65a527d3679b6d24f8fce0d1","changes":
{"changePasswordOnNextLogin":true},"action":"user props changing","details":
{"login":"dgu_user"}}
```

Ошибка при аутентификации:

```
CEF:0|ROSSINNO|SAYMON|3.12.86|AAF|Authorization error|8|dproc=saymon-server end=1709195078104
outcome=Failure suser= src=10.0.2.2 shost=_gateway dst=127.0.0.1 dhost=127.0.0.2 cn3=3
reason="User not found"  msg={"remoteAddress":"10.0.2.2","login":"badUser","message":"User not
found"}
```

Требование изменить пароль при входе:

```
CEF:0|ROSSINNO|SAYMON|3.12.86|AAF|Authorization error|8|dproc=saymon-server end=1709195096722
outcome=Failure suser=dgu_user src=10.0.2.2 shost=_gateway dst=127.0.0.1 dhost=127.0.0.2 cn3=3
reason="Password change is required."  msg={"errorCode":31,"message":"Password change is
required."}
```

Изменение пароля пользователем:

```
CEF:0|ROSSINNO|SAYMON|3.12.86|PCH|Password change by user|3|dproc=saymon-server
end=1709195114624 outcome=Success suser=dgu_user src= shost= dst= dhost= cs1=Password  msg=
{"entityType":"User","entityId":"65a527d3679b6d24f8fce0d1","changes":
{"passwordHash":"**","changePasswordOnNextLogin":false},"action":"user props
changing","details":{"login":"dgu_user"}}
```

7. Расширения

7.1. Расширения

Центральный Пульт поддерживает установку расширений, дополняющих функционал системы.

7.1.1. 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 Add или + на странице с расширениями. Также, список новых расширений можно найти на странице market.saymon.tech.

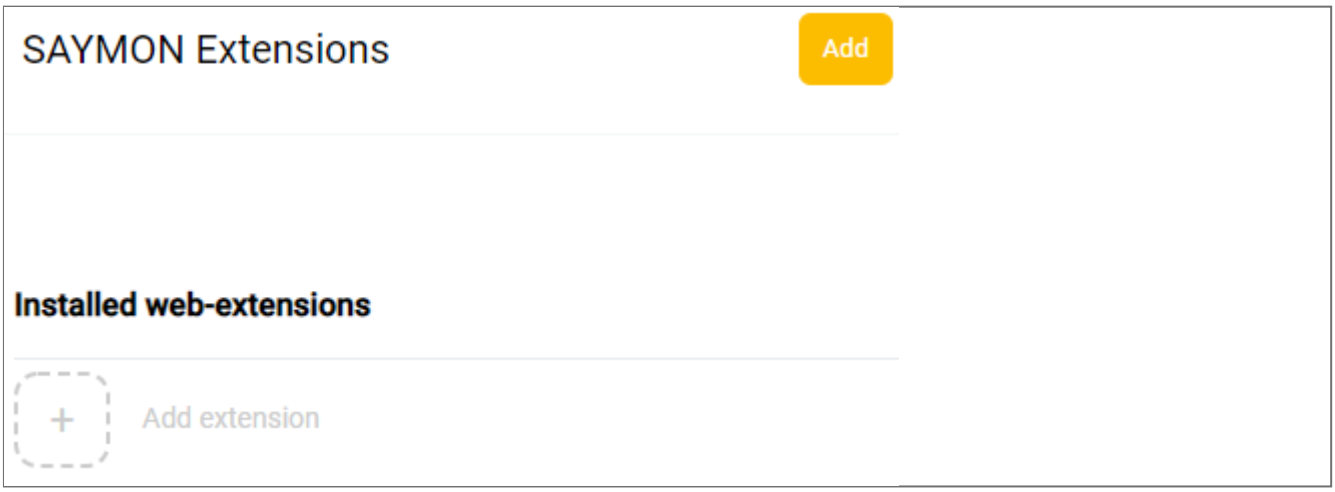


Рис. 9. Меню расширений

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

7.1.2. Пользовательские расширения

Пользователи могут писать собственные web-приложения, расширяющие или использующее возможности системы Центральный Пульт.

Чтобы включить такие приложения в состав сервера системы Центральный Пульт, 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>/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 системы Центральный Пульт, 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>/extensions/<nameInUrl>`
permissions	Набор пользовательских прав, которое добавляет расширение.
permissions.*<имя-права>*.description	Описание конкретного права в разделе конфигурации web-interface:configuration/users-and-groups-management/permissions.html. В этом поле можно указать разные описания для каждого поддерживаемого системой языка.
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",
    "permissions": {
      "custom_permission": {
        "description": {
          "ru": "Название прав пользователя",
          "en": "Permission name"
        }
      }
    }
}
```

8. Интеграции

8.1. Интеграция с Zabbix

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

8.1.1. Настройка подключения к Zabbix 5.x

Для настройки подключения нужно выполнить следующие шаги:

8.1.1.1. 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}
```

8.1.1.2. 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.

8.1.1.3. 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.

8.1.1.4. 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
    }
]
```

8.1.1.5. Шаг 5: применить настройки

Для применения настроек подключения к серверу Zabbix необходимо перезапустить сервер системы Центральный Пульт:

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

8.1.2. Настройка подключения к Zabbix 6.0

WARNING

Интеграцию с Zabbix 6.0 поддерживает только сервер Центральный Пульт, 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 системы Центральный Пульт с Zabbix 6.0, выполните следующие действия:

8.1.2.1. 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
```

8.1.2.2. 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
```

8.1.2.3. 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:

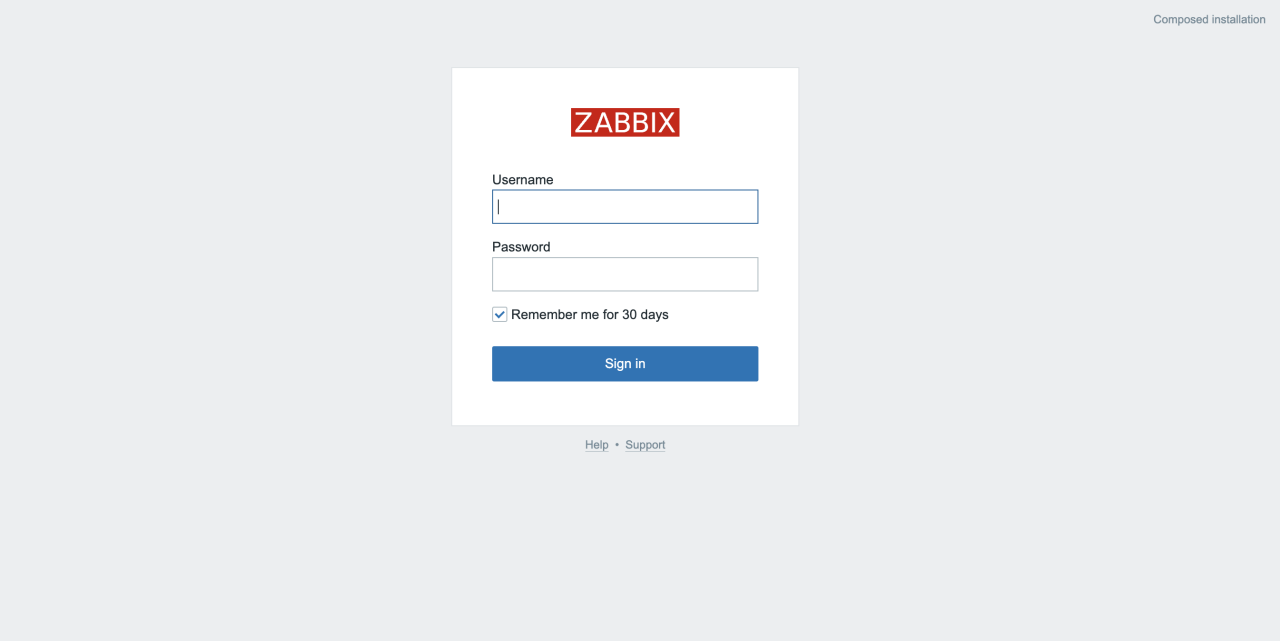
```
$ sudo systemctl status zabbix-agent
```

Остановка сервиса:

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

8.1.2.4. Шаг 4: подключение Zabbix Agent

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



NOTE

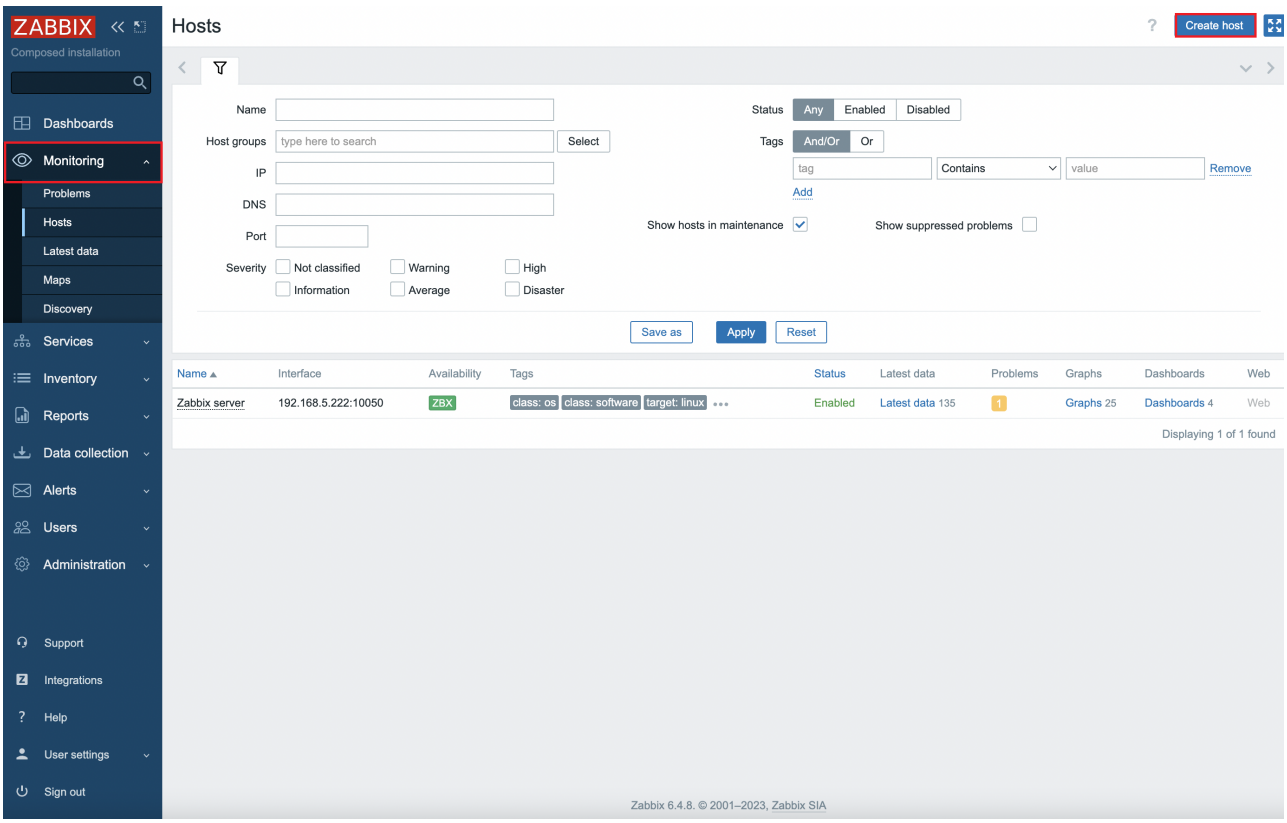
По умолчанию:

Логин: Admin

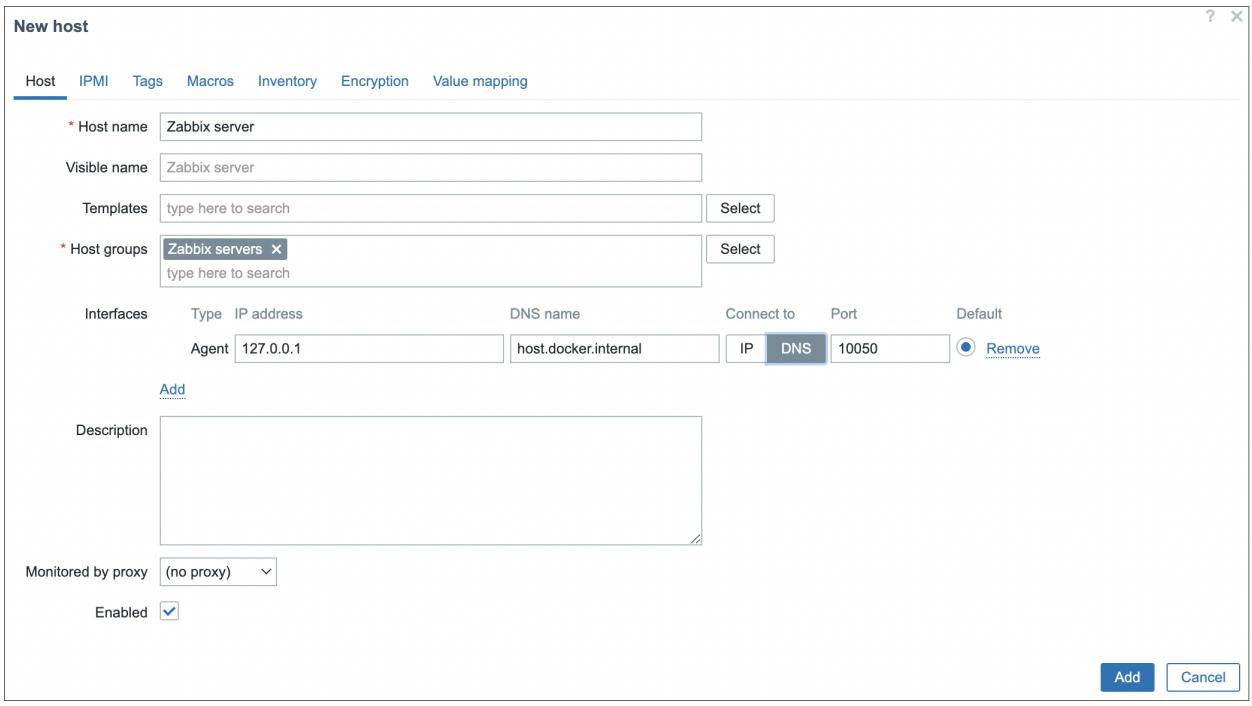
Пароль: zabbix

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.



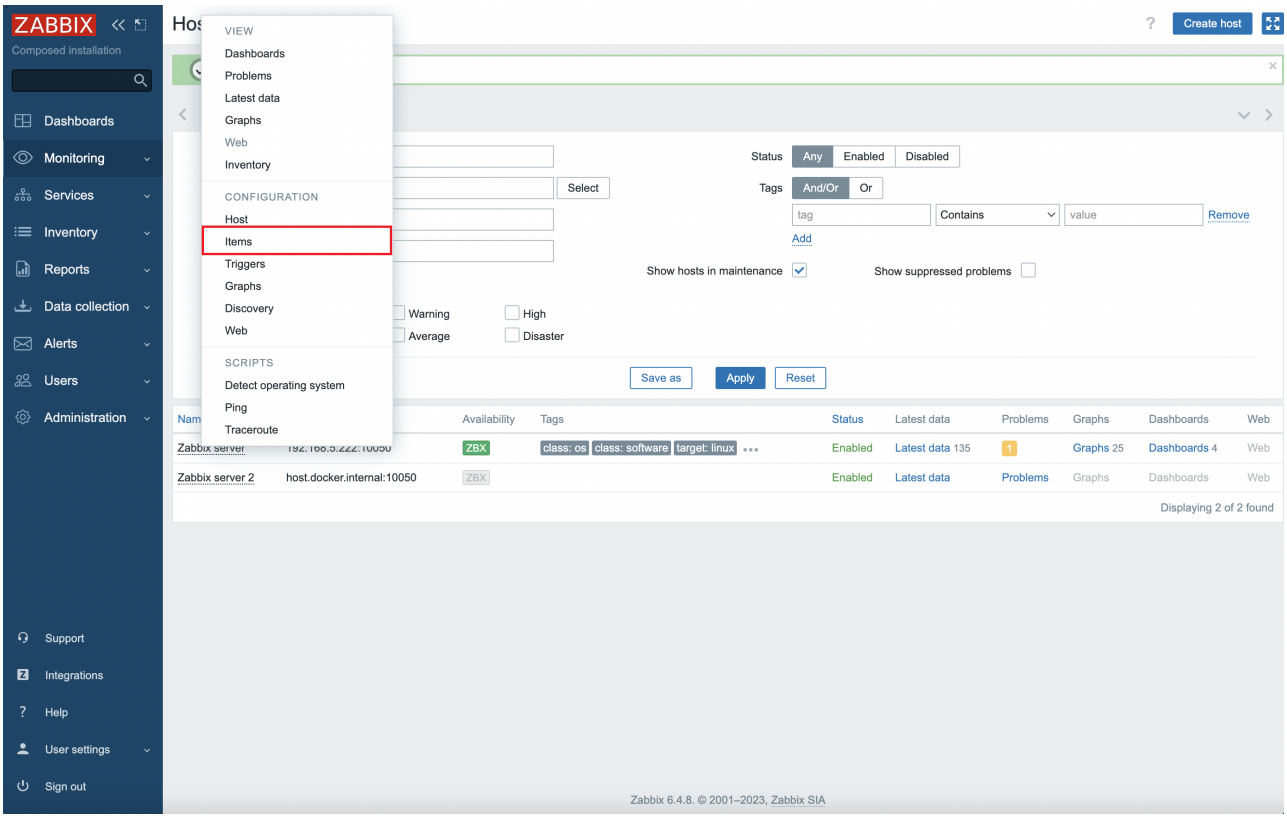
4. Нажмите кнопку **Add**.

В результате сервер Zabbix будет получать данные с агента.

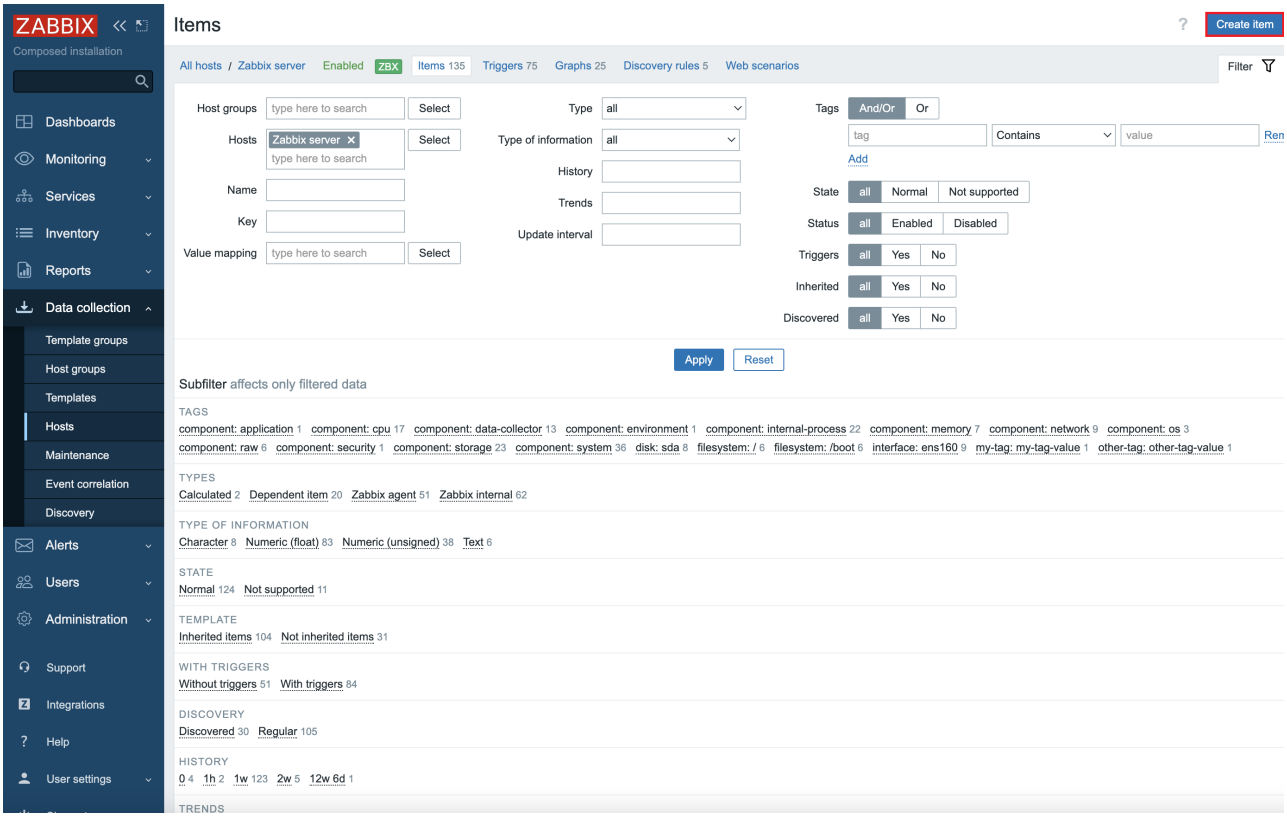
8.1.2.5. Шаг 5: создание объекта для мониторинга

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

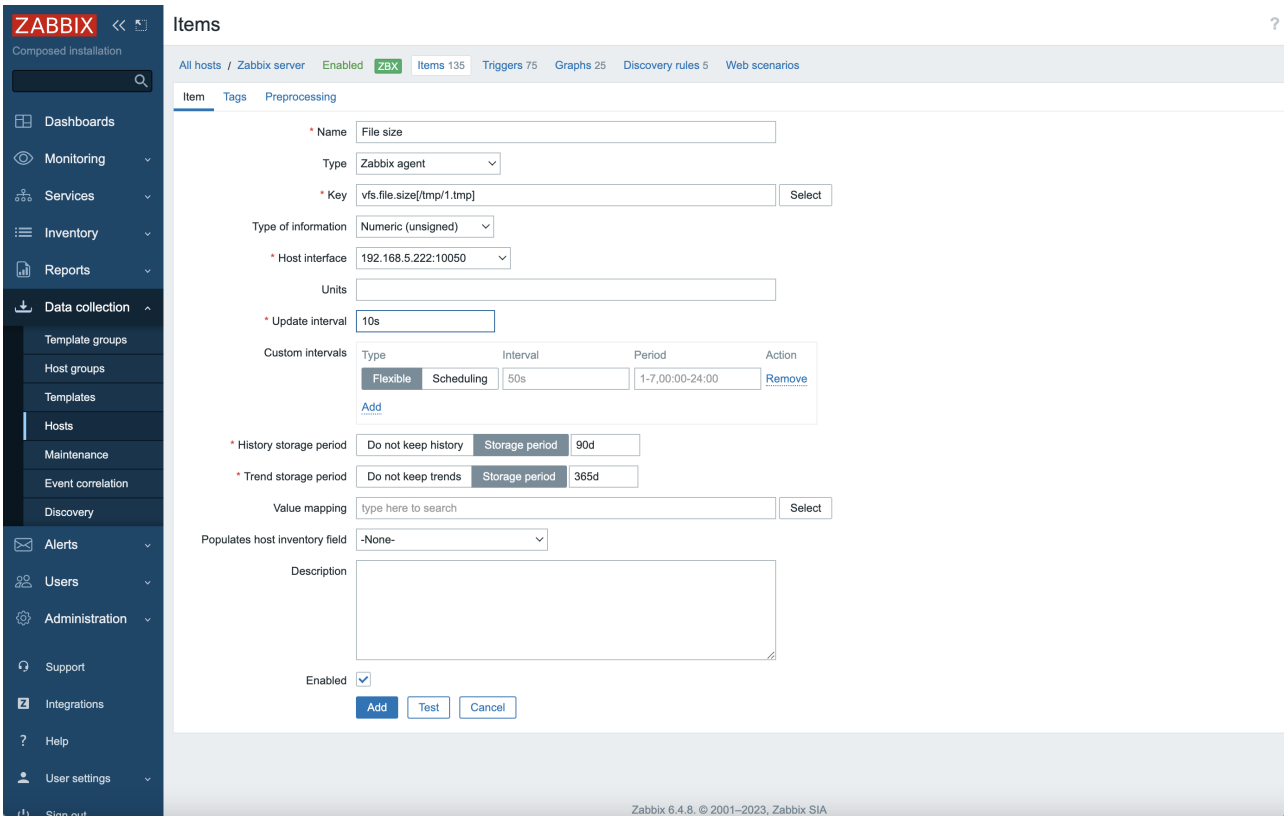
1. Нажмите на имя созданного сервера и в открывшемся меню выберите **Items**.



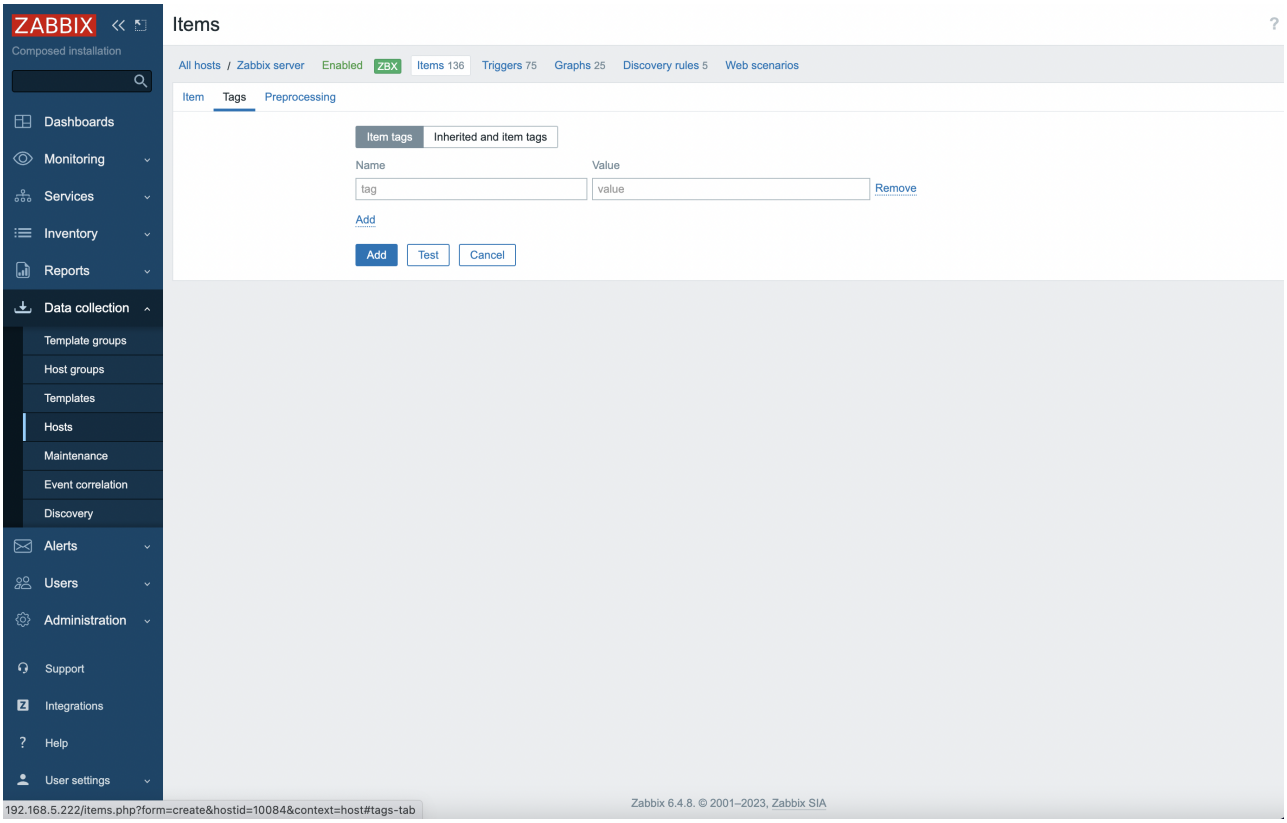
2. Нажмите кнопку **Create item** в правом верхнем углу станицы.



В открывшемся диалоговом окне задайте нужные параметры для item.



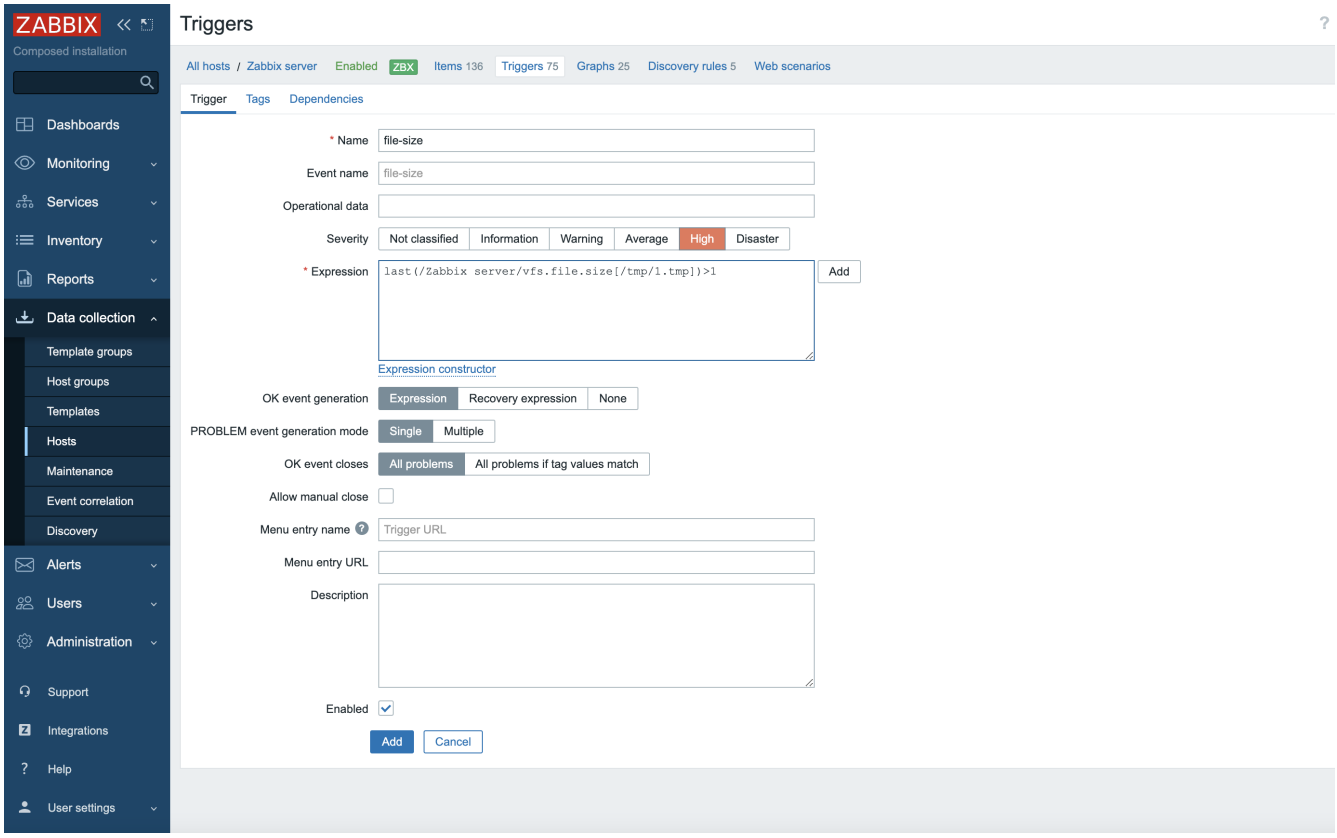
4. Переключитесь на вкладку **Tags**, задайте нужные теги и нажмите кнопку **Add**.



8.1.2.6. 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.

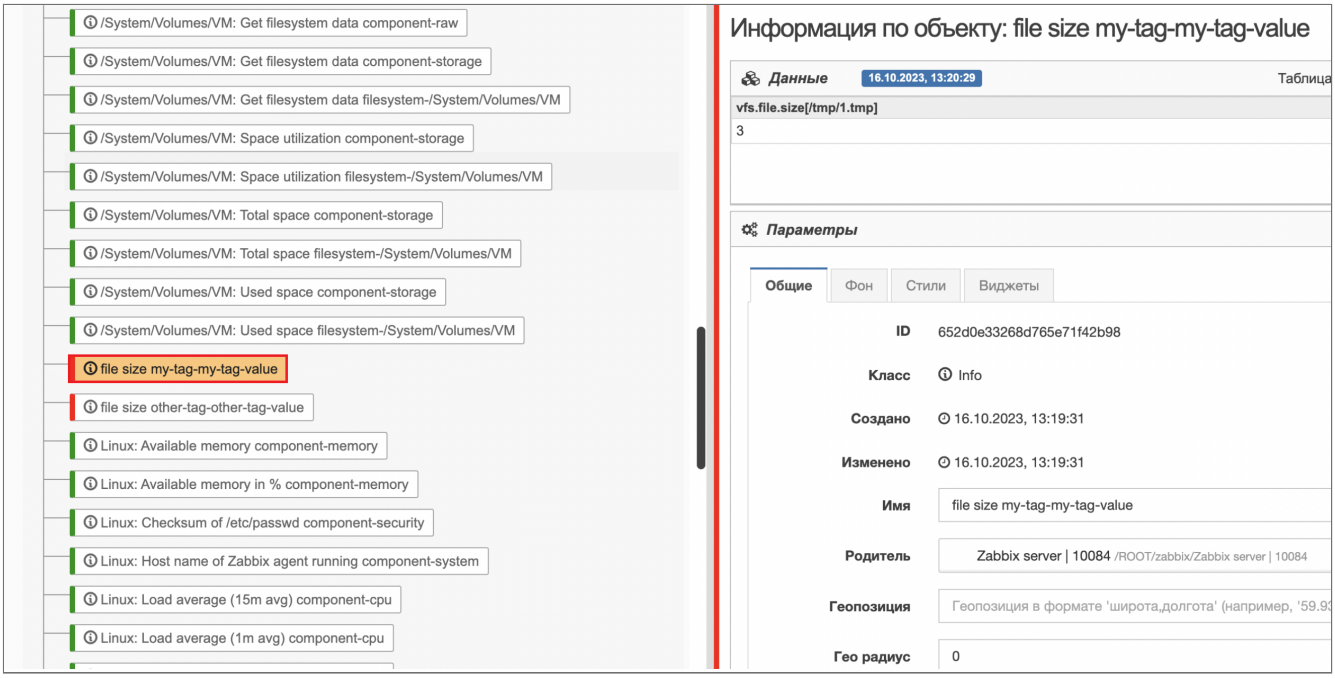


8.1.2.7. Шаг 7: настройка подключения к серверу "Центральный Пульт"

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

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

В списке объектов системы Центральный Пульт отобразятся Zabbix-хост и его дочерние объекты.



8.1.2.8. 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
```

8.1.3. 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 системы Центральный Пульт:

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

8.1.4. 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
     }
...
}]
```

8.1.5. 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"

8.1.6. Журналирование

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

```
"zabbix" : [{
   ...
   "debug": true,
   ...
   }
]
```

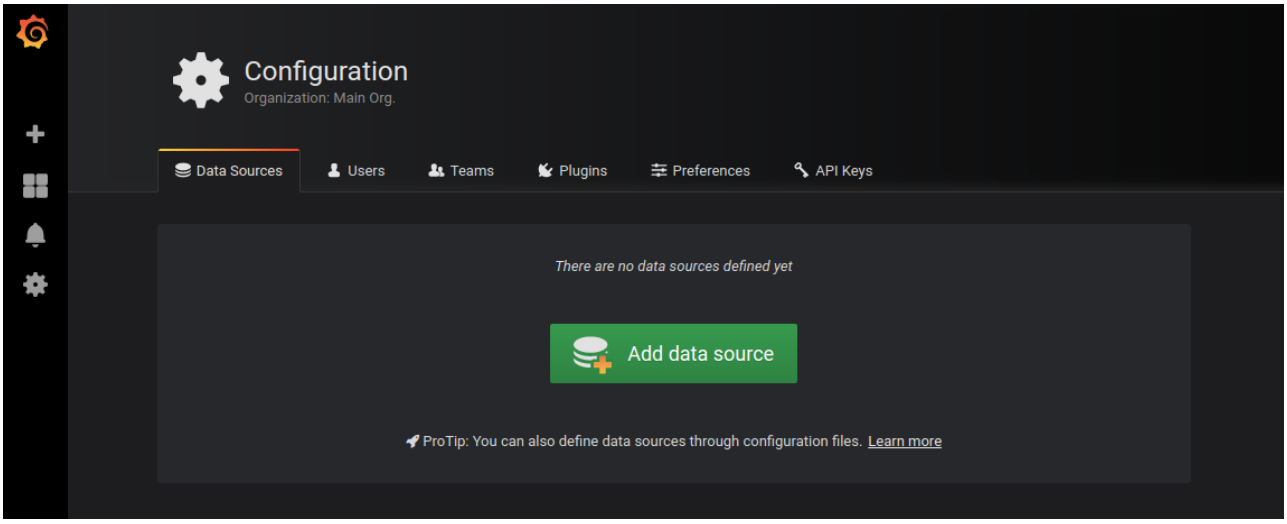
8.2. Интеграция с Grafana

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

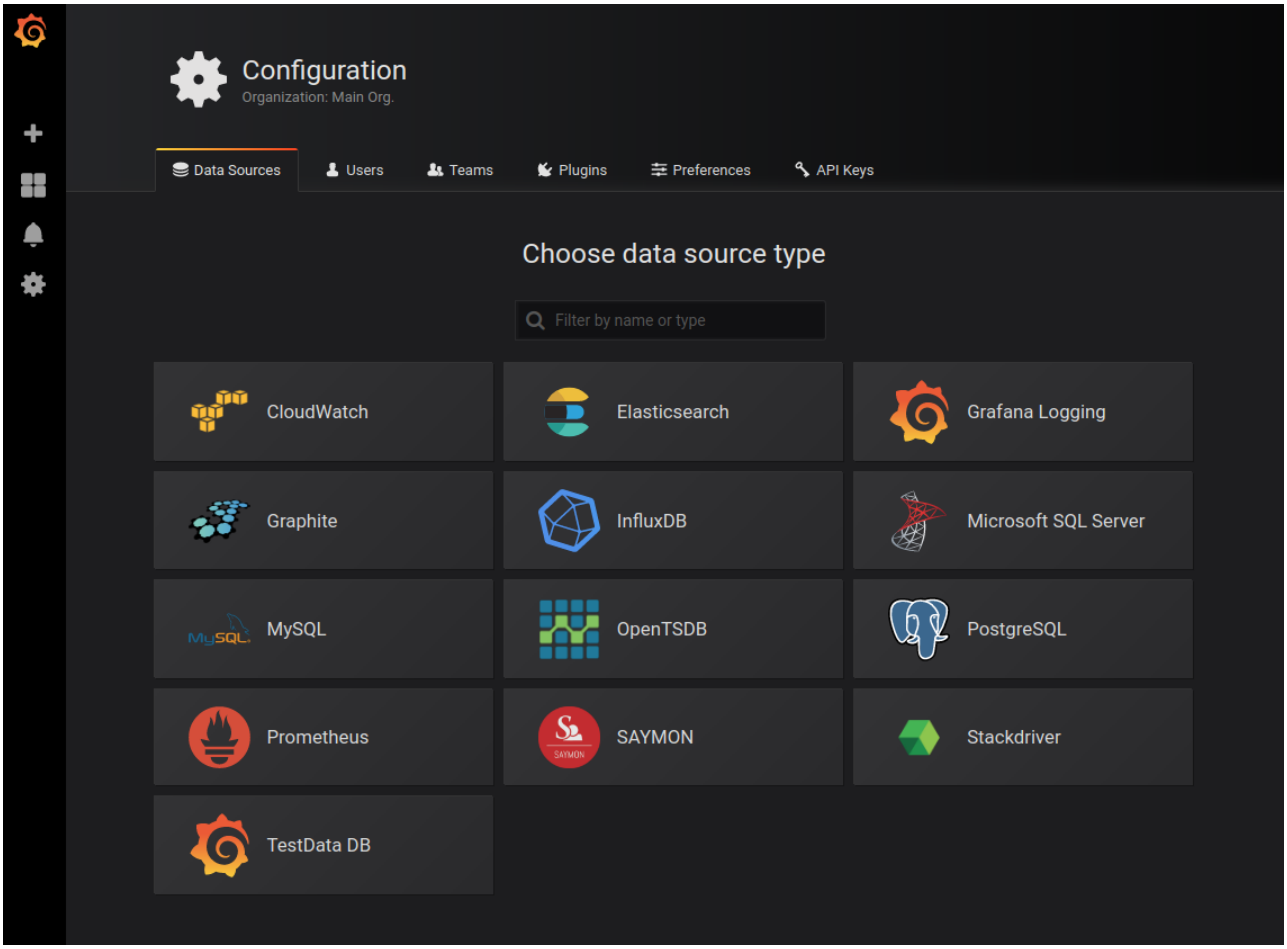
Центральный Пульт интегрируется с Grafana через специальный datasource-плагин, который позволяет отображать на диаграммах данные по указанным метрикам из указанных объектов системы Центральный Пульт. При помощи стандартного инструментария Grafana данные визуализируются, и на их основании строятся дашборды.

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

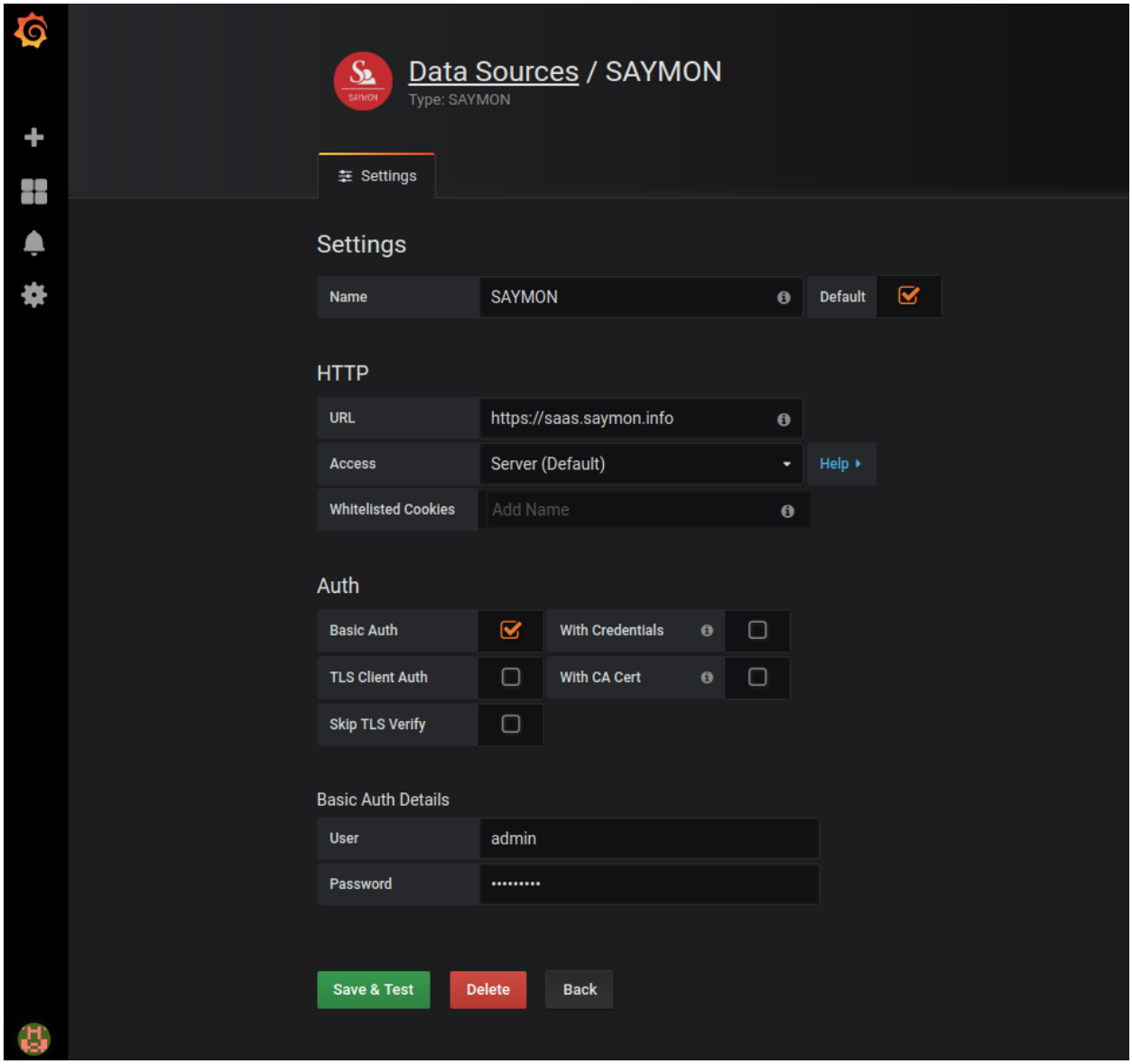
1. Авторизоваться в Grafana.
2. С главной страницы рабочего пространства перейти в **Configurations - Data Sources** через боковое меню и добавить data source (источник данных) нажатием на соответствующую кнопку.



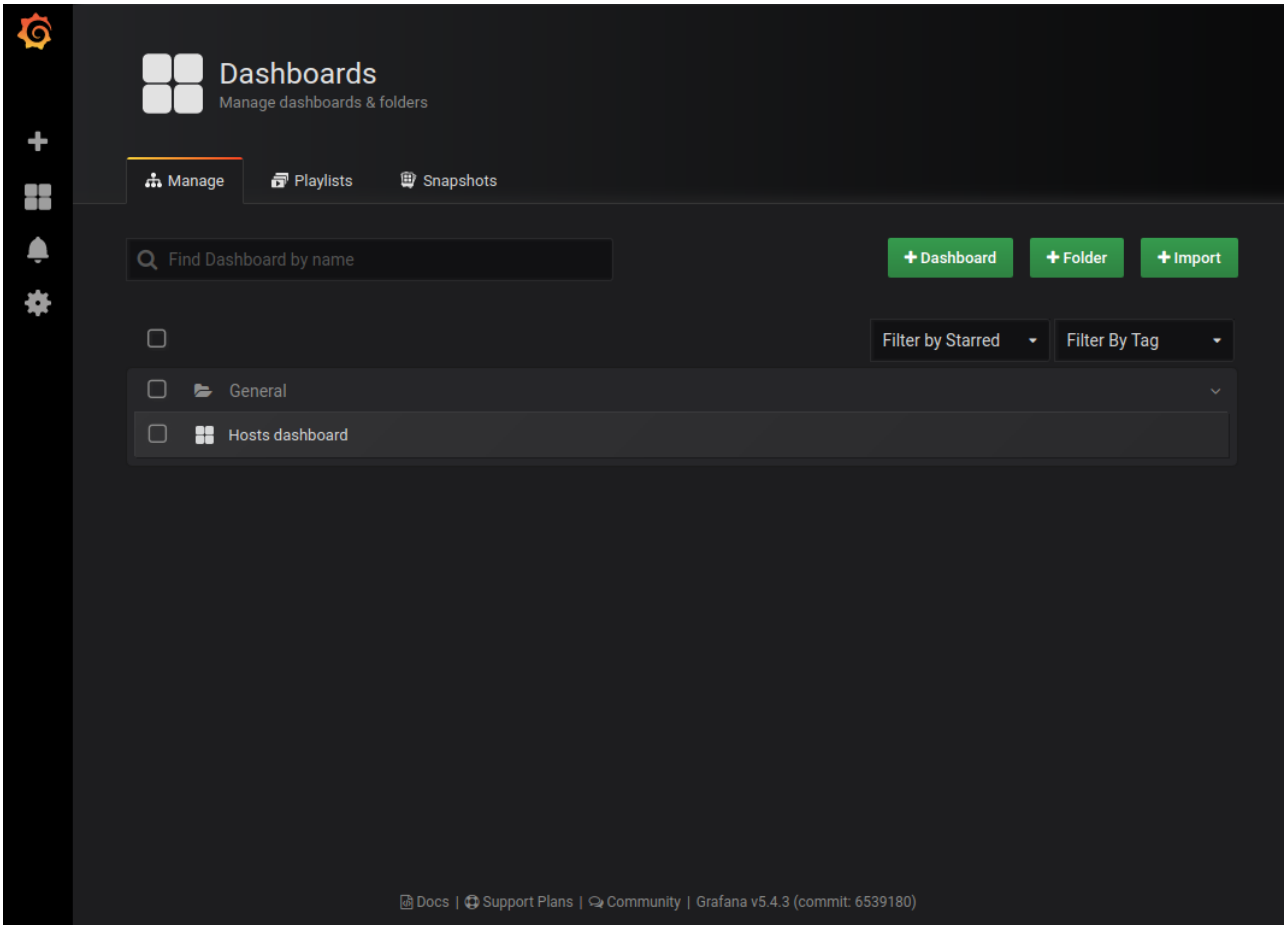
3. В открывшемся окне типов источников данных выбрать SAYMON.



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 системе Центральный Пульт.



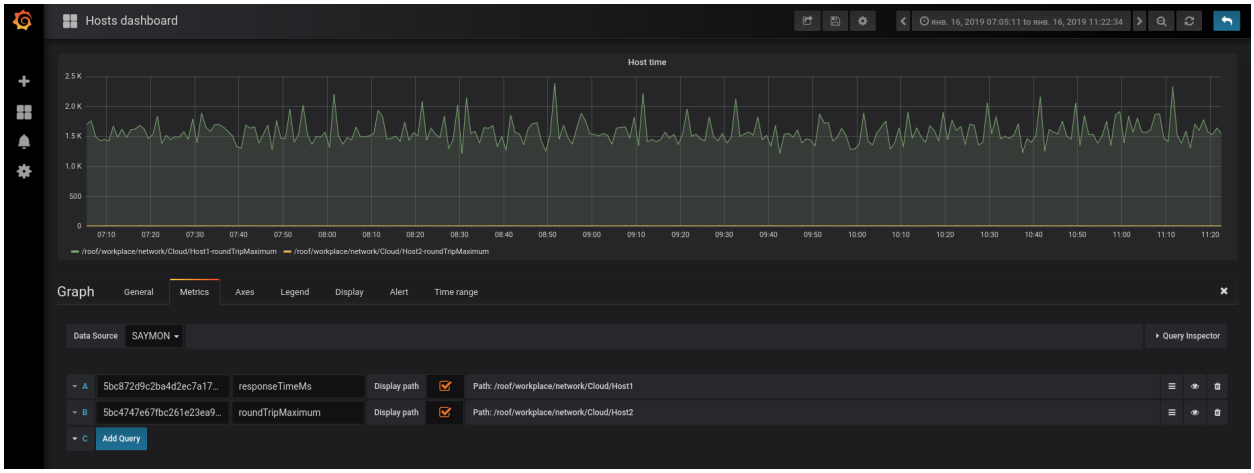
5. Через боковое меню перейти в окно управления дашбордами (**dashboards - manage**) и приступить к созданию нового дашборда нажатием соответствующей кнопки.



6. Выбрать тип отображения дашборда. Например, график.



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 системе Центральный Пульт. Указать:
   - имя или ID объекта;
   - метрику, которую необходимо отобразить на графике;
   - отображать ли путь до объекта.



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

8.3. Интеграция с Keycloak

Центральный Пульт поддерживает авторизацию посредством сервера Keycloak.

TIP

Поддерживается интеграция только с Keycloak версии 21.1.1.

8.3.1. Настройка сервера

Для настройки взаимодействия между сервером системы Центральный Пульт 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.

8.3.1.1. Realm Certificate

Для получения Realm Certificate необходимо:

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

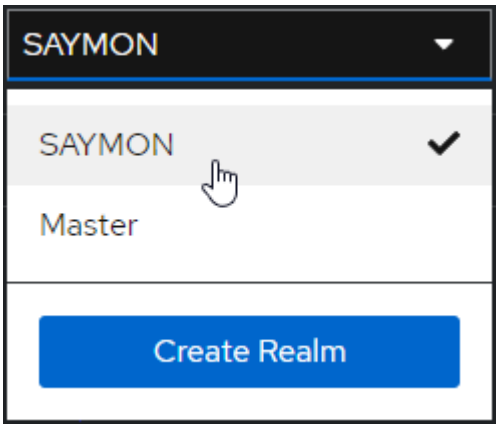


Рис. 10. Выбор области в Keycloak

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

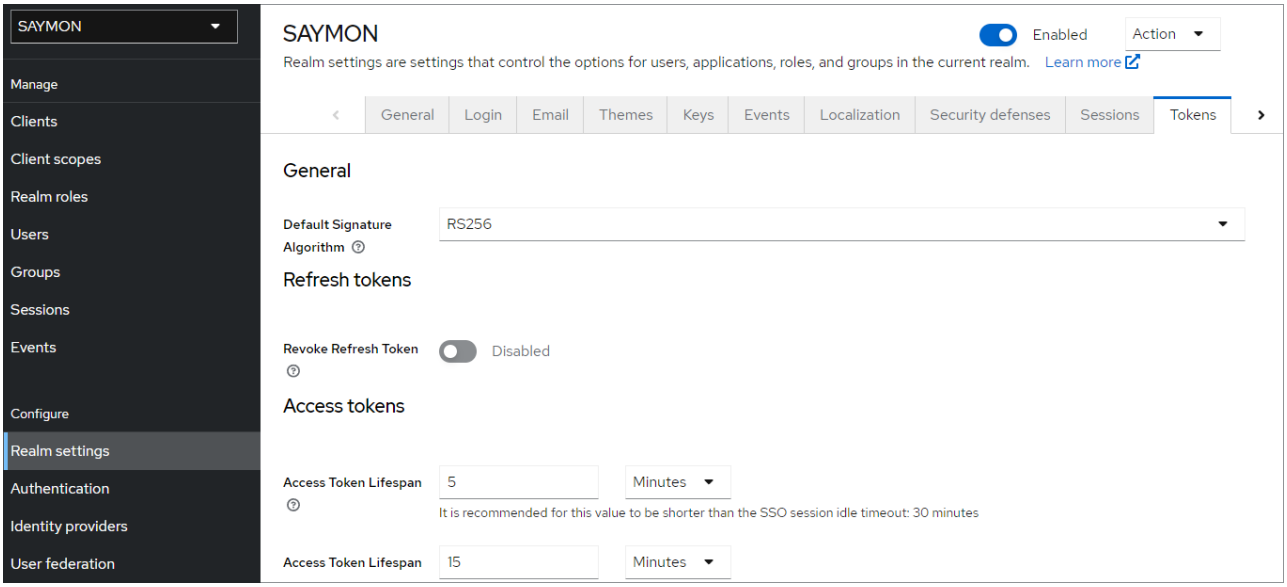


Рис. 11. Выбор алгоритма подписи токенов

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

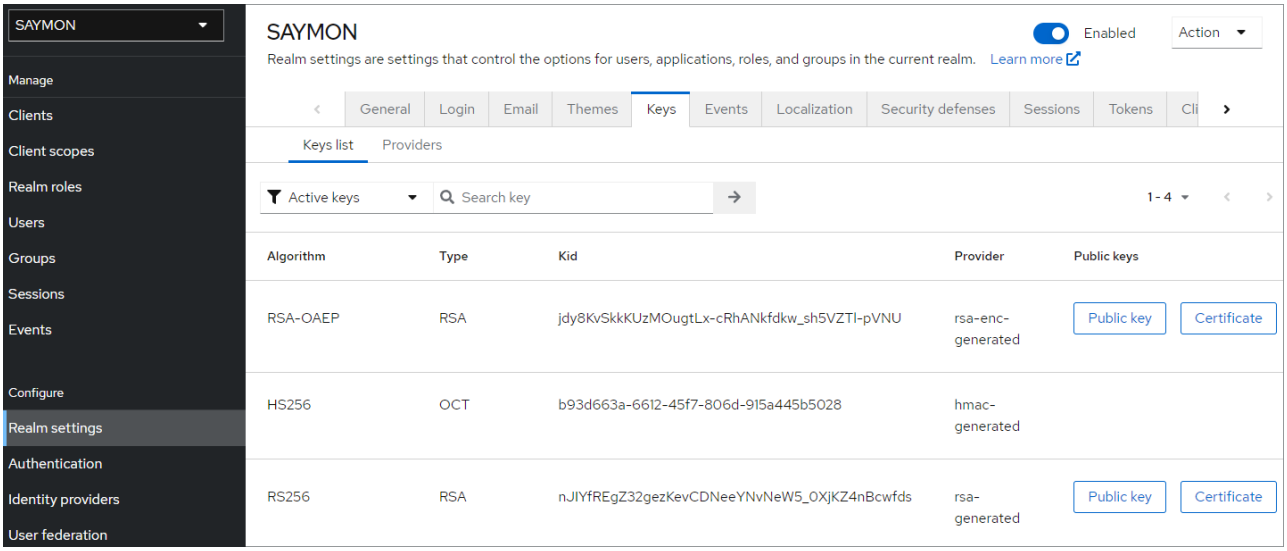


Рис. 12. Вкладка Keys

4. Кнопка Certificate в соответствующей строке открывает искомый сертификат:

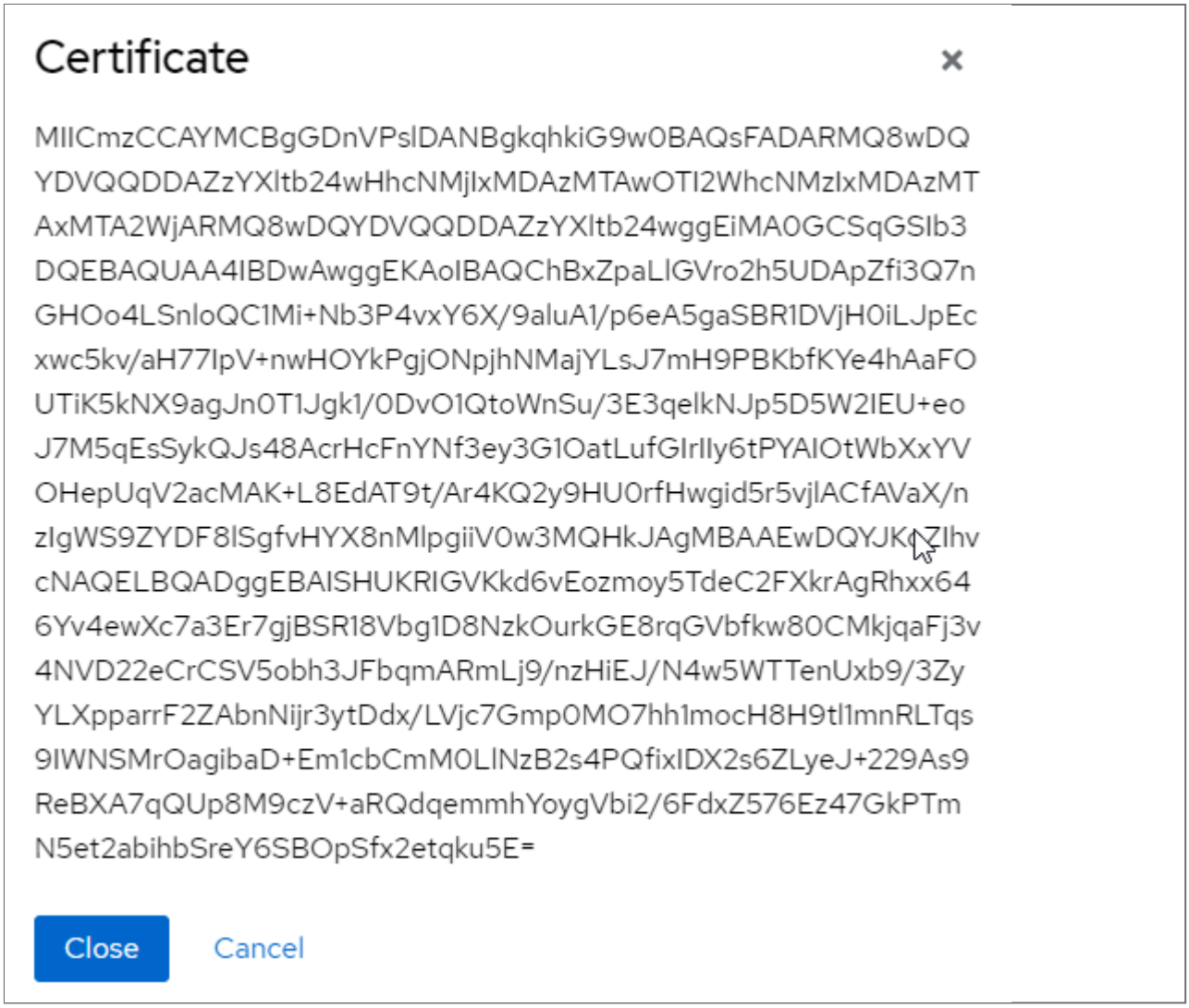


Рис. 13. Сертификат ключа на вкладке Keys

Данные рекомендуется сохранить в файл **keycloak_realm_certificate**.

8.3.1.2. 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**):

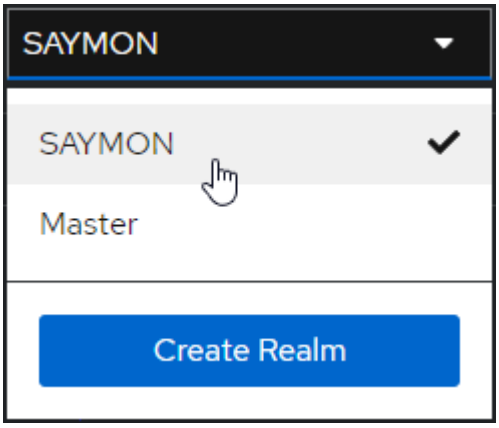


Рис. 14. Выбор области в Keycloak

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

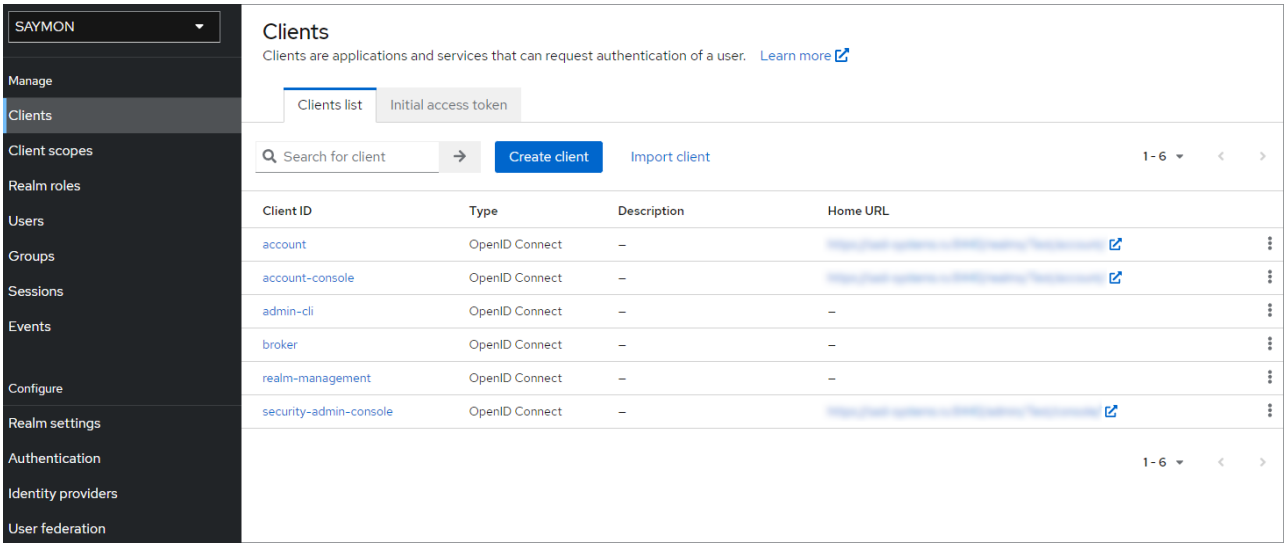


Рис. 15. Секция Clients

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

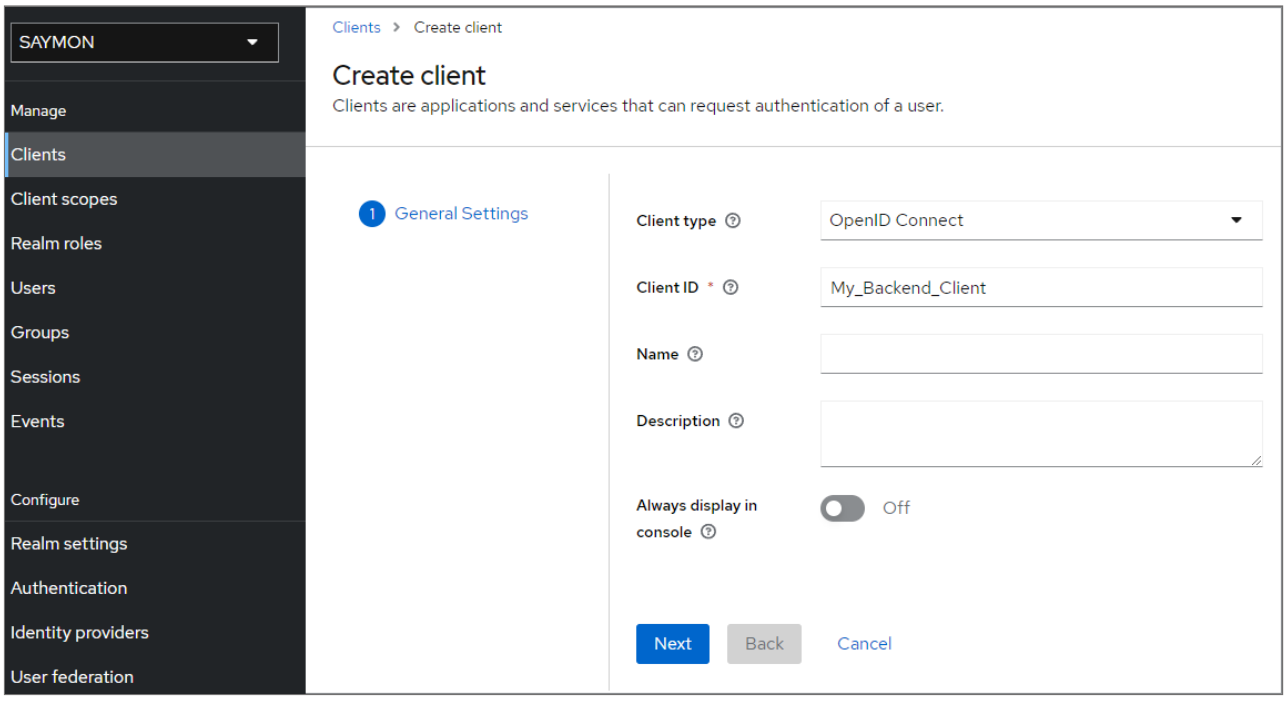


Рис. 16. Создание клиента, вкладка General Settings

и нажать кнопку **Next**.

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

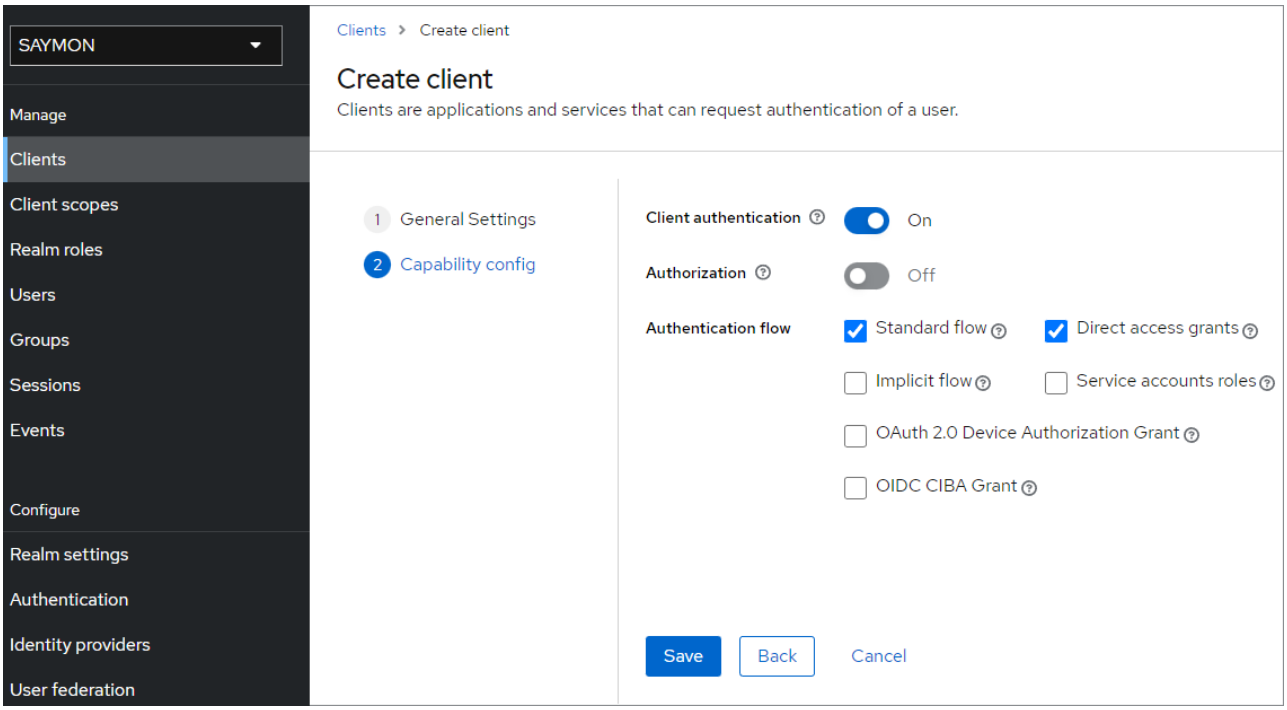


Рис. 17. Создание клиента, вкладка Capability config

и нажать кнопку **Save**.

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

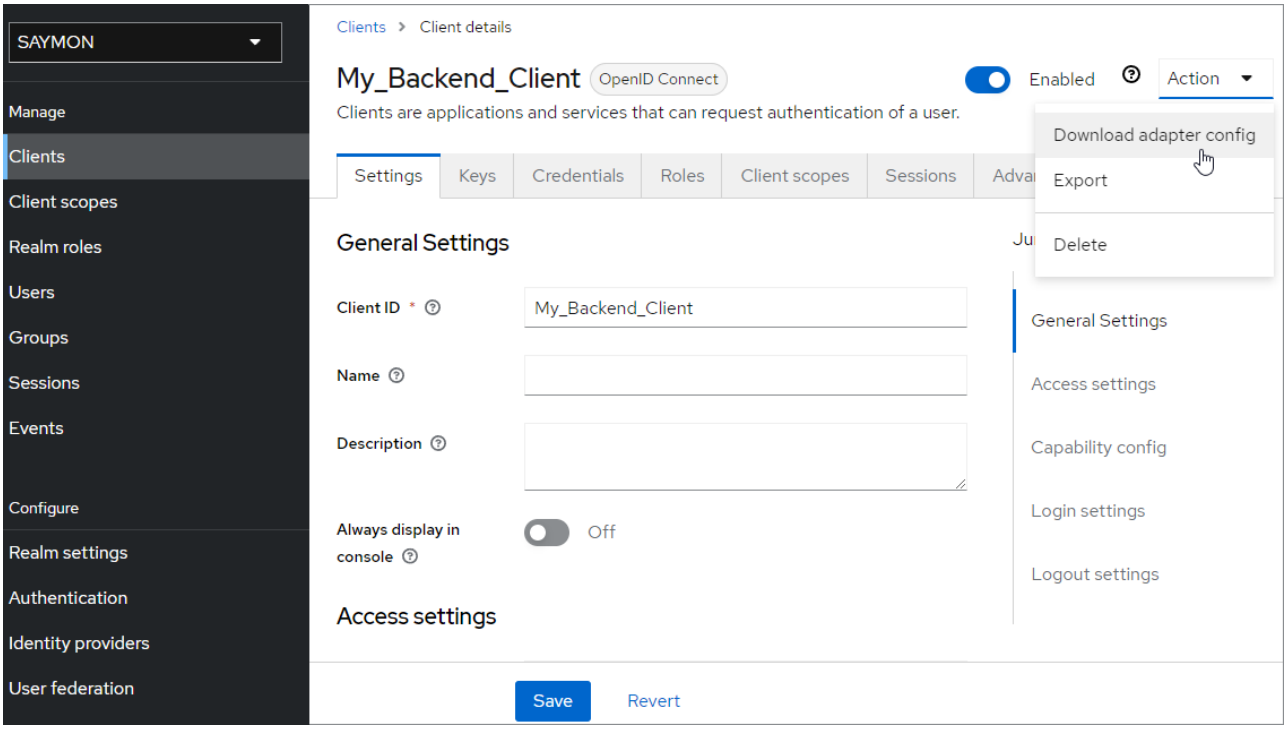


Рис. 18. Кнопка Download adapter config

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

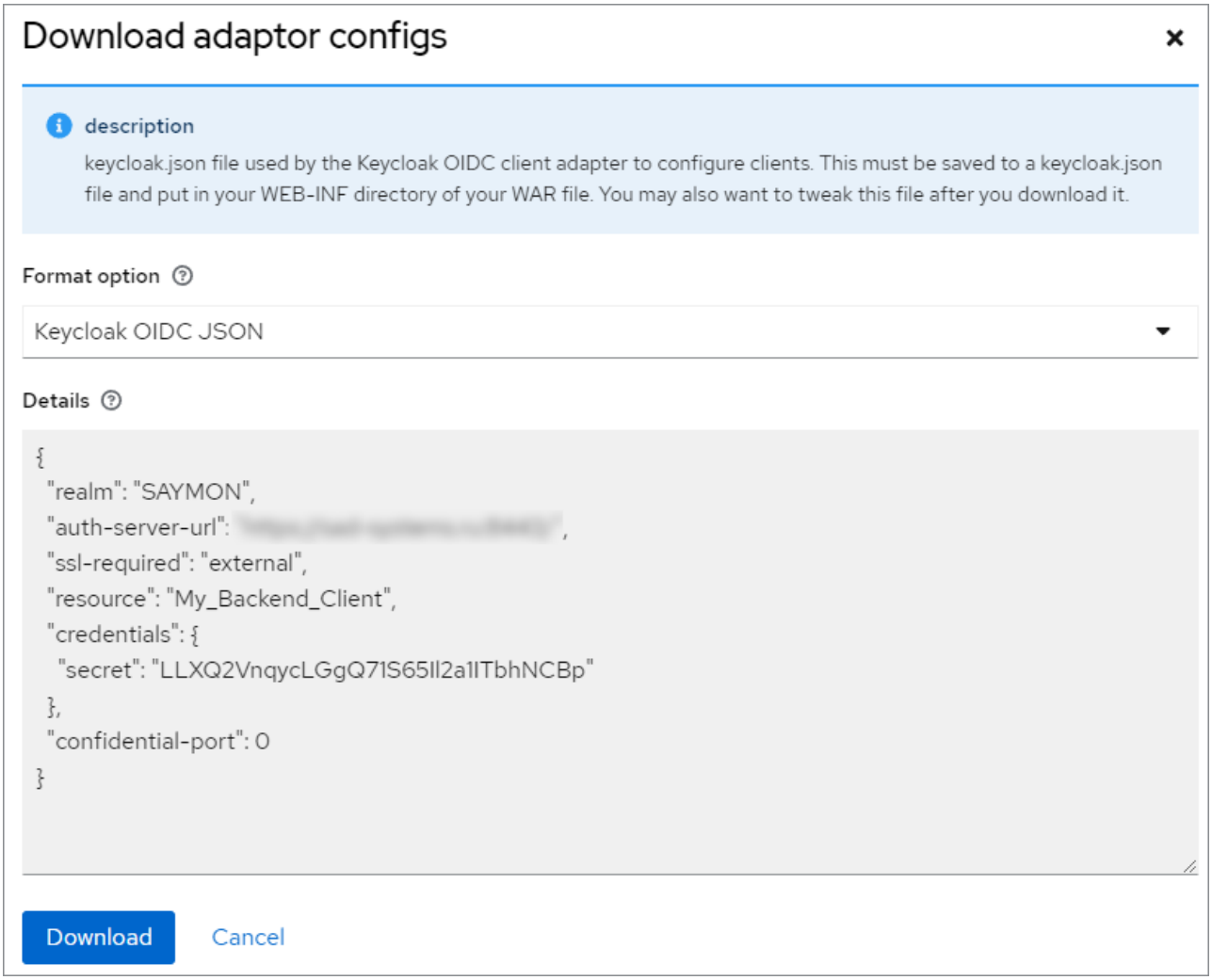


Рис. 19. Файл настроек соединения для Backend-клиента

8.3.1.3. Конфигурация сервера Центральный Пульт

Для взаимодействия с сервером Keycloak на сервере "Центральный Пульт" 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
```

8.3.2. Настройка web-интерфейса

Для возможности авторизации пользователей посредством Keycloak через web-интерфейс системы Центральный Пульт необходимо получить файл настроек соединения Frontend-клиента.

WARNING

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

8.3.2.1. 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**):

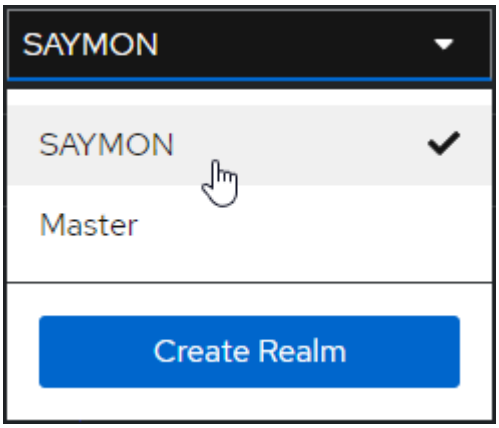
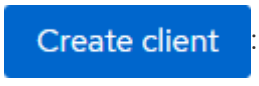


Рис. 20. Выбор области в Keycloak

2. В секции **Clients** нажать кнопку 

:

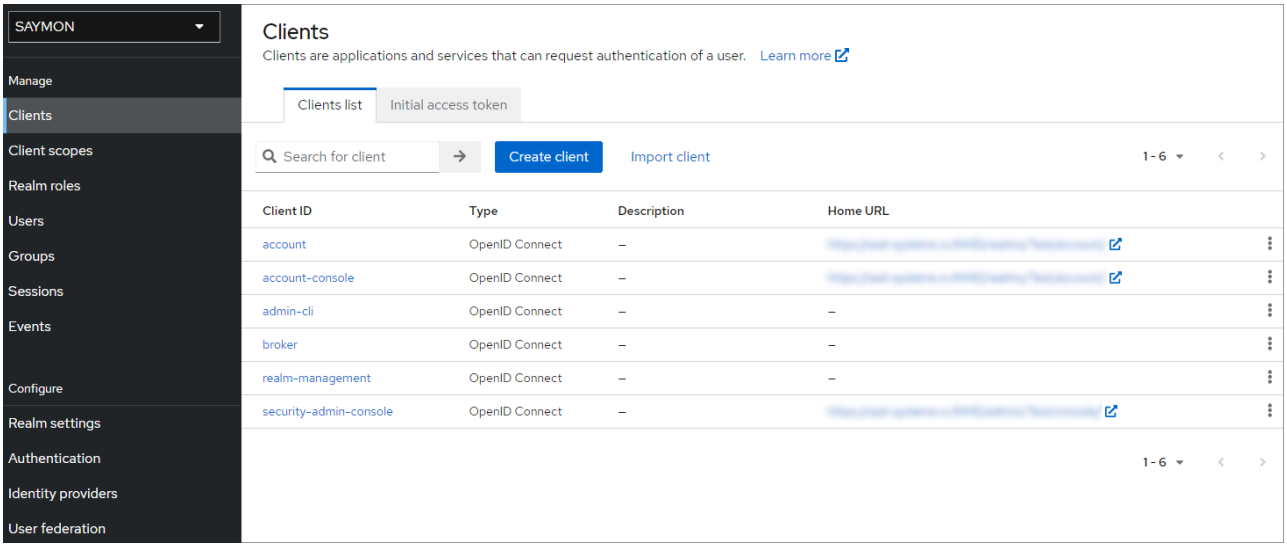


Рис. 21. Секция Clients

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

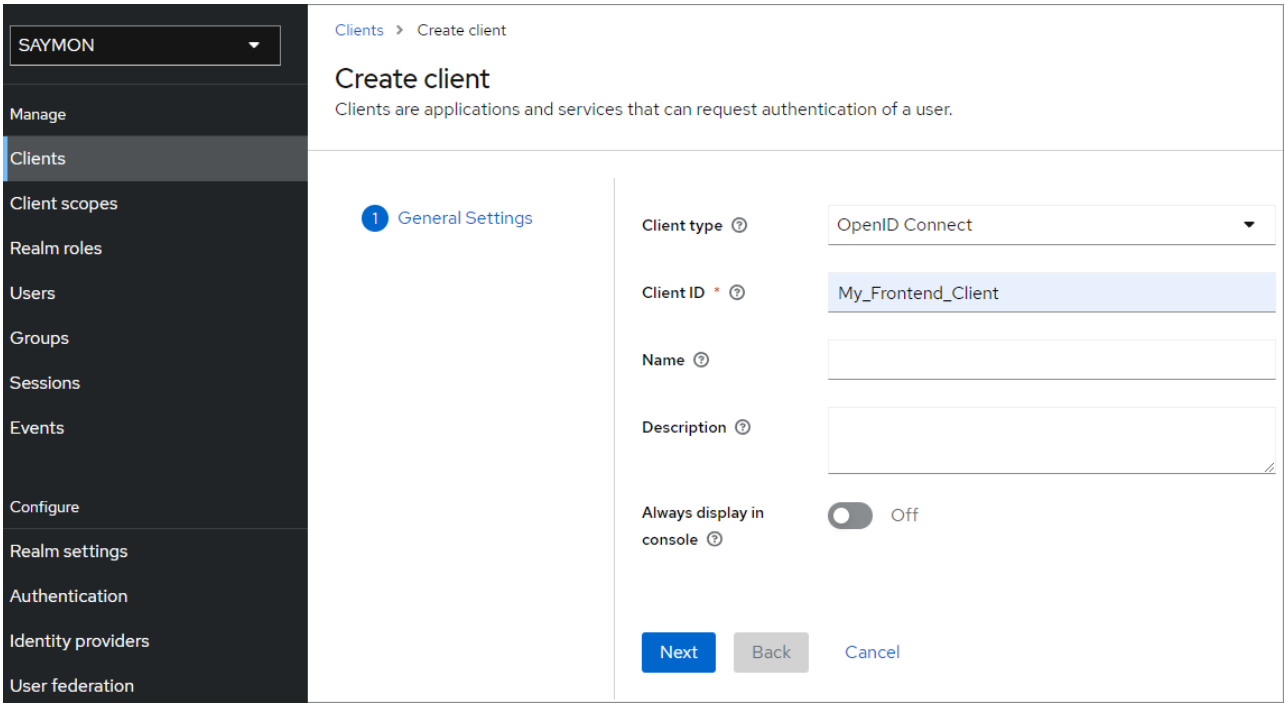


Рис. 22. Создание клиента, вкладка General Settings

и нажать кнопку Next.

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

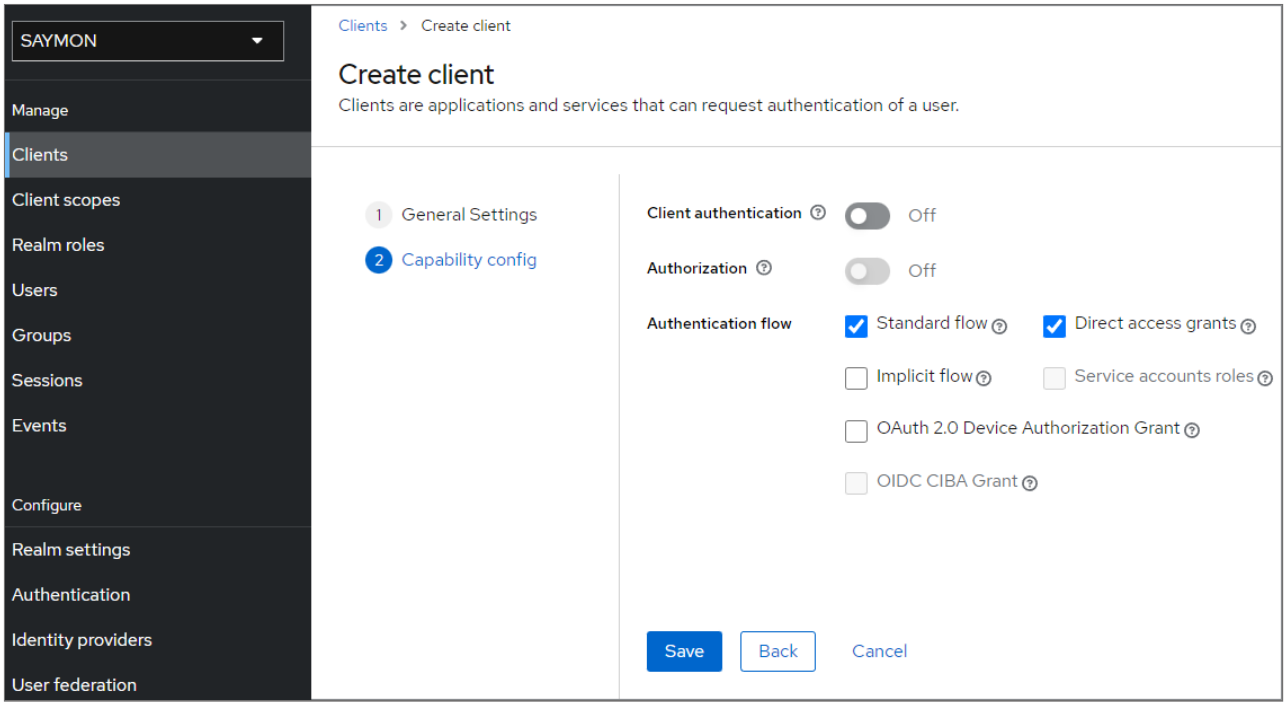


Рис. 23. Создание клиента, вкладка Capability config

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:

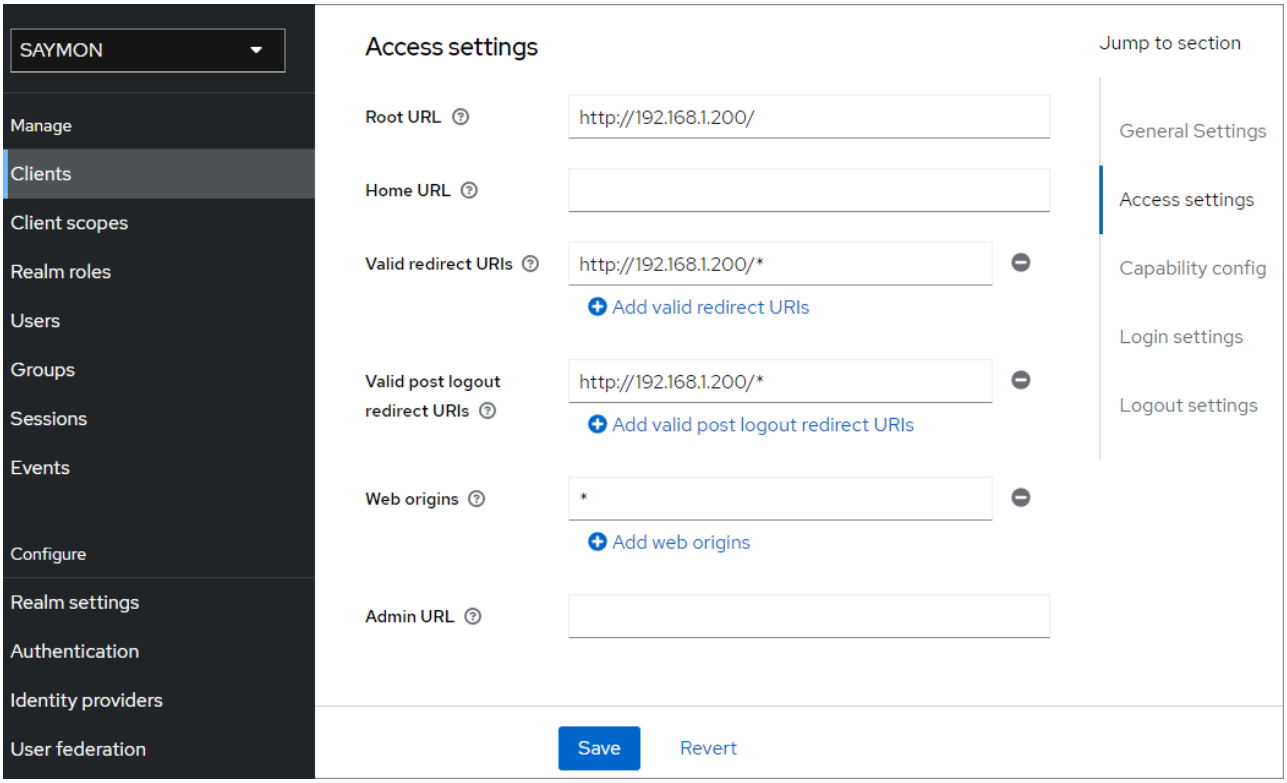


Рис. 24. Раздел Access Settings

и нажать кнопку **Save**.

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

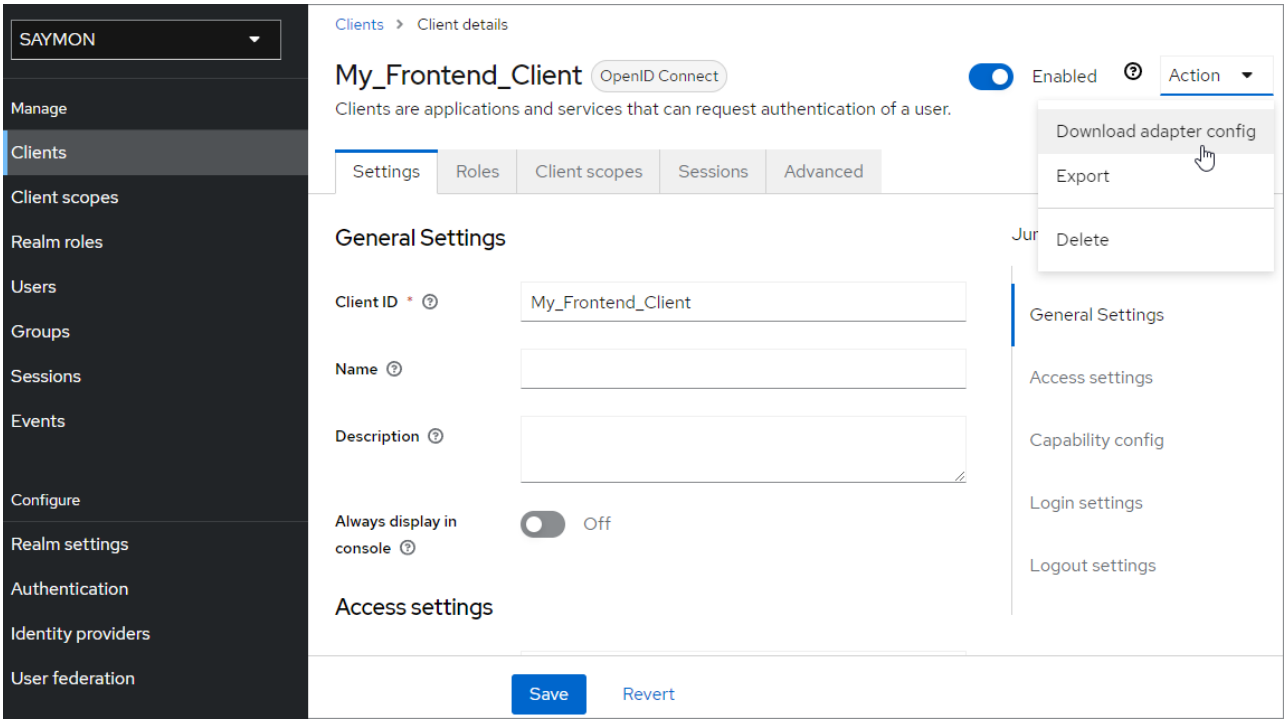


Рис. 25. Кнопка Download adapter config

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

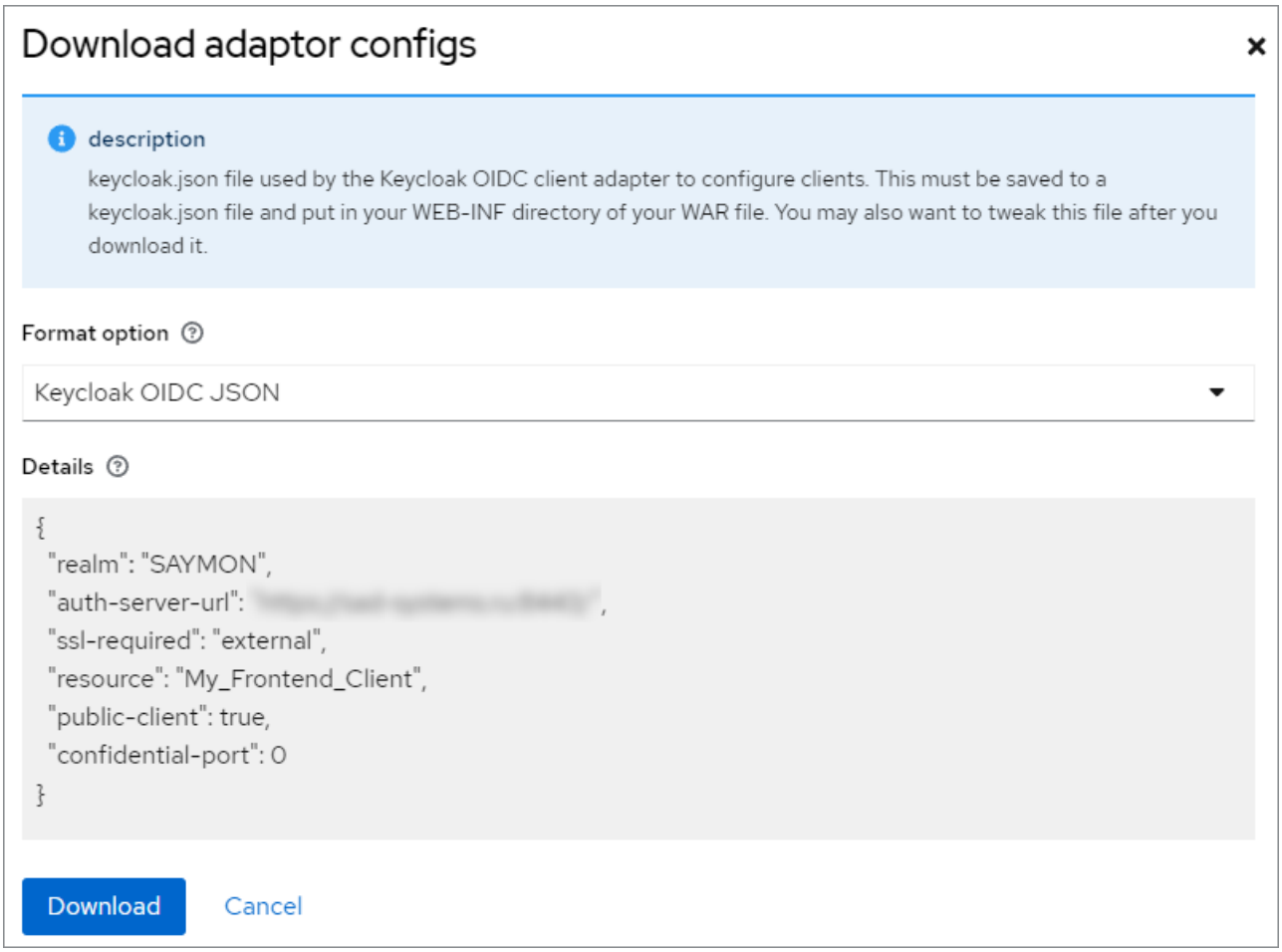


Рис. 26. Файл настроек соединения для Frontend-клиента

8.3.2.2. Конфигурация web-интерфейса системы Центральный Пульт

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

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

   ```
   authentication:
     availableMethods:
       - native
       - keycloak
     methodByDefault: keycloak
     services:
       keycloak:
         configUri: saymon.local/keycloak_frontend.json
         legacySupport: false
   ```

   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 системы Центральный Пульт в web-браузере.

8.4. Интеграция с VictoriaMetrics

Чтобы настроить хранение метрик объектов в VictoriaMetrics, необходимо выполнить следующие действия:

8.4.1. Шаг 1: Установка VictoriaMetrics

С помощью указанных ниже команд можно загрузить последний доступный образ VictoriaMetrics и запустить его через порт **8428**, сохраняя при этом полученные данные в подкаталоге **victoria-metrics-data** текущего каталога:

```
$git clone https://github.com/VictoriaMetrics/VictoriaMetrics && cd VictoriaMetrics
$docker pull victoriametrics/victoria-metrics:stable
$docker run -it --rm -v `pwd`/victoria-metrics-data:/victoria-metrics-data -p 8428:8428
victoriametrics/victoria-metrics:stable
```

NOTE

Для корректной работы необходимо выполнить запуск VictoriaMetrics с открытым портом **8428**.

8.4.2. Шаг 2: Настройка конфигурации сервера

Для настройки конфигурации сервера необходимо добавить в файл серверной конфигурации (`/etc/saymon/saymon-server.conf`) секцию **tsdb_extensions**, которая представляет собой массив конфигураций дополнительных баз временных рядов:

```
"tsdb_extensions": [
  {
    "name": "victoria",
    "enabled": true,
    "config": {
      "promAddr": "http://<vitoriametrics_host>:8428"
    }
  }
]
```

Описание параметров:

- **name** - имя БД (необходимо использовать **victoria**);
- **enabled** - включена/выключена;
- **config** - конфигурация для подключения к БД с параметром **promAddr** до порта включительно.

NOTE

Если в секции **tsdb_extensions** заданы другие подключения, то для них необходимо задать **enabled: false**.

8.5. Интеграция с InfluxDB

Начиная с версии 3.6.80, Центральный Пульт поддерживает хранение метрик объектов в InfluxDB.

Для этого необходимо выполнить следующие действия:

8.5.1. Шаг 1: Установка InfluxDB

Для загрузки докер-образа InfluxDB можно использовать следующие команды:

```
$git clone https://hub.docker.com/_/chronograf && cd chronograf
$docker pull chronograf
```

Для запуска InfluxDB можно использовать следующую команду:

```
$docker run -d -p 8086:8086 \
      -v $PWD:/var/lib/influxdb \
      --name=influxdb \
      --net=influxdb \
      influxdb
```

При необходимости можно запустить веб-приложение Chronograf:

```
$docker run -p 8888:8888 \
      --net=influxdb \
      chronograf --influxdb-url=http://influxdb:8086
```

NOTE

Доступ к веб-интерфейсу InfluxDB будет осуществляться по адресу `localhost:8888`

8.5.2. Шаг 2: Настройка конфигурации сервера

Для настройки конфигурации сервера необходимо добавить в файл серверной конфигурации `/etc/saymon/saymon-server.conf` секцию **tsdb_extensions**, которая представляет собой массив конфигураций дополнительных баз временных рядов:

```
   "openTsdb" : {
        "host" : "localhost",
        "port" : 4242
    },
    "tsdb_extensions": [
        {
            "name": "influx",
            "enabled": true,
            "config": {}
        }
    ],
```

NOTE

В целях обратной совместимости содержимое секции **OpenTsdb** рекомендуется оставить без изменений.

Описание элементов массива:

- **name** - имя БД (для InfluxDB используется имя **influx**);
- **enabled** - включена/выключена;
- **config** - конфигурация для подключения к БД.
- **path** - абсолютный путь до файла клиента БД. Используется в том случае, если требуется подключить ещё какой-либо сторонний модуль для работы с временными рядами.

Подключение будет устанавливаться к первой указанной базе данных, у которой значение поля `enabled` равно `true`. Если для всех указанных БД параметр enabled равен `false`, то клиент будет подключаться к БД OpenTSDB.